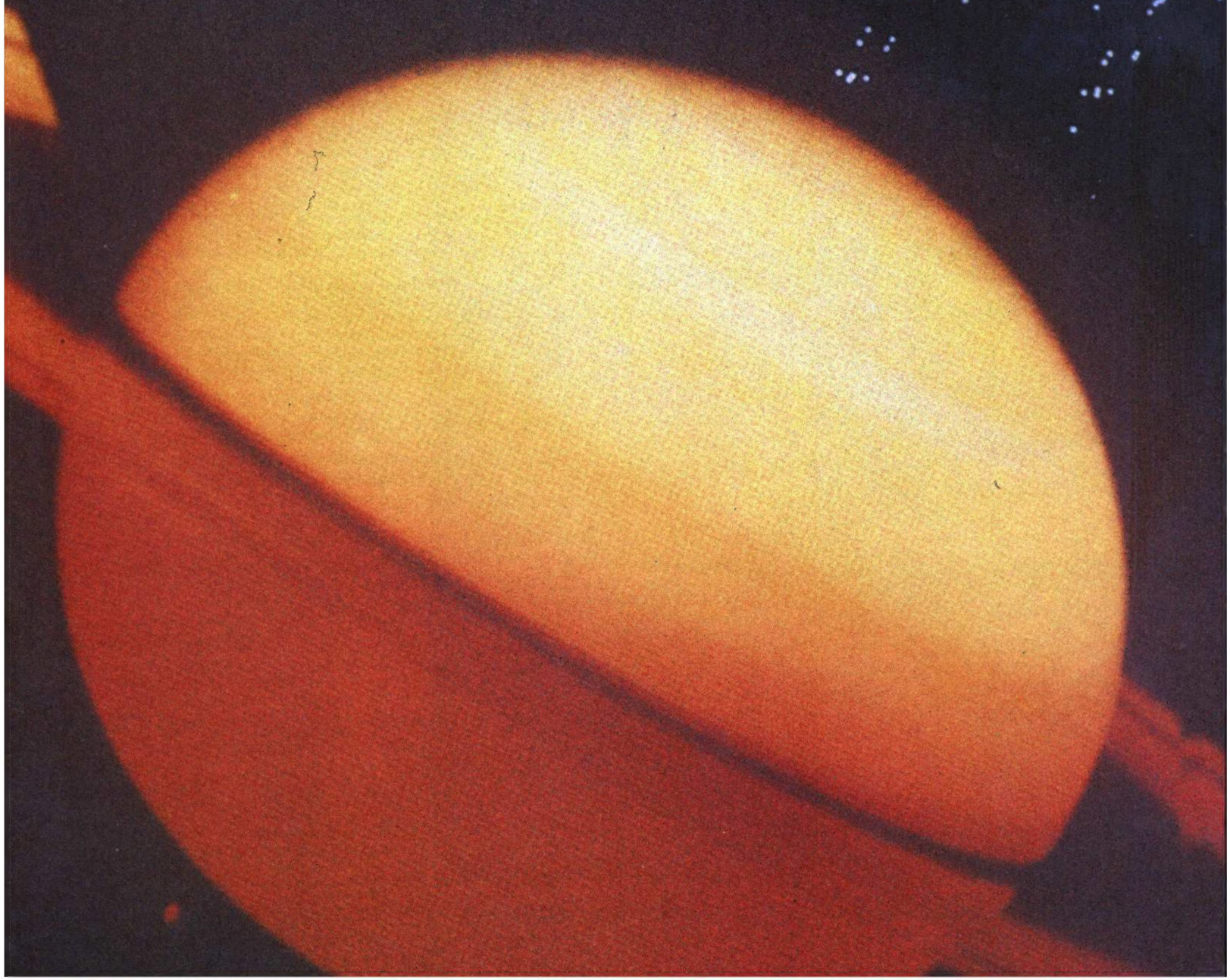
CURSO PRÁTICO 59 DE PROGRAMAÇÃO DE COMPUTADORES

INTERNET

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

NOVA CULTURAL

Cz$ 50,00



INPUT

Vol. 4 N.º 59

NESTE NÚMERO

PROGRAMAÇÃO BASIC

PADRÕES NATURAIS

A utilização do microcomputador na reprodução de padrões da natureza. Famílias de curvas. Envoltórias. Focos e cúspides. Teoria das catástrofes. Padrões formados por pontos. Simulação do fenômeno da ressonância. Caos. Dinâmica de populações 1161

CÓDIGO DE MÁQUINA

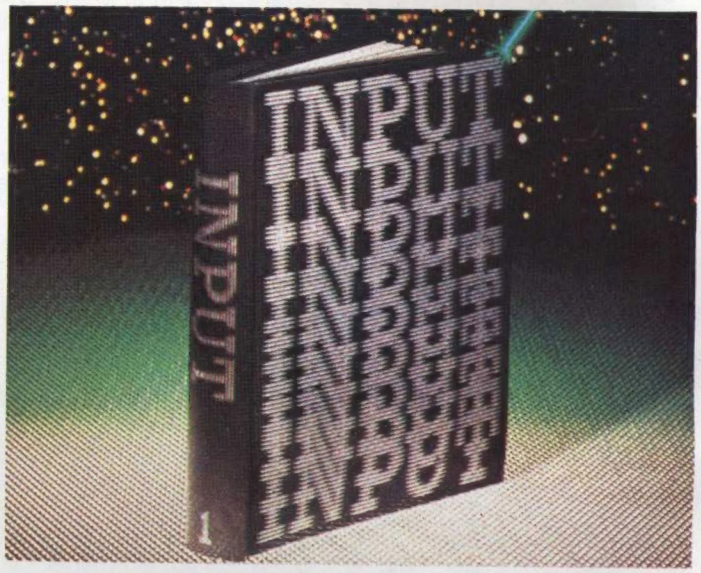
***AVALANCHE:* OS SALTOS DE WILLIE**

As quatro partes da rotina de salto. Salto vertical e salto diagonal. Efeitos sonoros que acompanham a rotina. Ajustamento dos apontadores de tela. Willie está subindo. A figura no ar. De volta ao solo. Fim do salto 1168

PROGRAMAÇÃO BASIC

MODELOS DA REALIDADE

Modelos icônicos, análogos e simbólicos. Incertezas. Comportamento aleatório e probabilidade. Variáveis aleatórias. Simulações. Distribuição uniforme. Distribuição normal. Distribuição exponencial. Comparação de eventos 1176



PLANO DA OBRA

"INPUT" é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: 1. PESSOALMENTE — Por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em **São Paulo**, os endereços são: rua Brigadeiro Tobias, 773, Centro; avenida Industrial, 117, Santo André; e no **Rio de Janeiro**: avenida Mem de Sá, 191/193, Centro. 2. POR CARTA — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidora Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132, Jardim Teresa — CEP 06000 — Osasco — SP. Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na agência do Correio. 3. POR TELEX — Utilize o nº (011) 33 670 DNAP.

Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações, Lda. — Qta. Pau Varais, Azinhaga de Fetais — 2 685, Camarate — Lisboa; Apartado 57 — Telex 43 069 JARLIS P.

Atenção: Após seis meses do encerramento da coleção, os pedidos serão atendidos dependendo da disponibilidade do estoque.

Obs.: Quando pedir livros, mencione sempre título e/ou autor da obra, além do número da edição.

COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao Serviço de Atendimento ao Leitor — Caixa Postal 9442, São Paulo — SP.

Editor
VICTOR CIVITA

REDAÇÃO
Diretor Editorial: Carmo Chagas

Editores Executivos: Antonio José Filho, Berta Sztark Amar

Editor Chefe: Paulo de Almeida
Editora de Texto: Ana Lúcia B. de Lucena
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Dagmar Bastos Sampaio, Grace Alonso Arruda, Monica Lenardon Corradi
Secretária de Redação/ Coordenadora: Stefania Crema
Secretários de Redação: Beatriz Hagström, José Benedito de Oliveira Damião, Maria de Lourdes Carvalho, Marisa Soares de Andrade, Mauro de Queiroz

COLABORADORES

Consultor Editorial Responsável: Dr. Renato M. E. Sabbatini (Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda., Campinas, SP
Tradução, adaptação, programação e redação: Abílio Pedro Neto, Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Marcos Huascar Velasco, Raul Neder Porrelli, Ricardo J. P. de Aquino Pereira
Coordenação Geral: Rejane Felizatti Sabbatini

COMERCIAL
Diretor Comercial: Roberto Martins Silveira
Gerente Comercial: Flávio Maculan
Gerente de Circulação: Denise Maria Mozol

PRODUÇÃO
Gerente de Produção: João Stungis
Coordenador de Impressão: Atílio Roberto Bonon

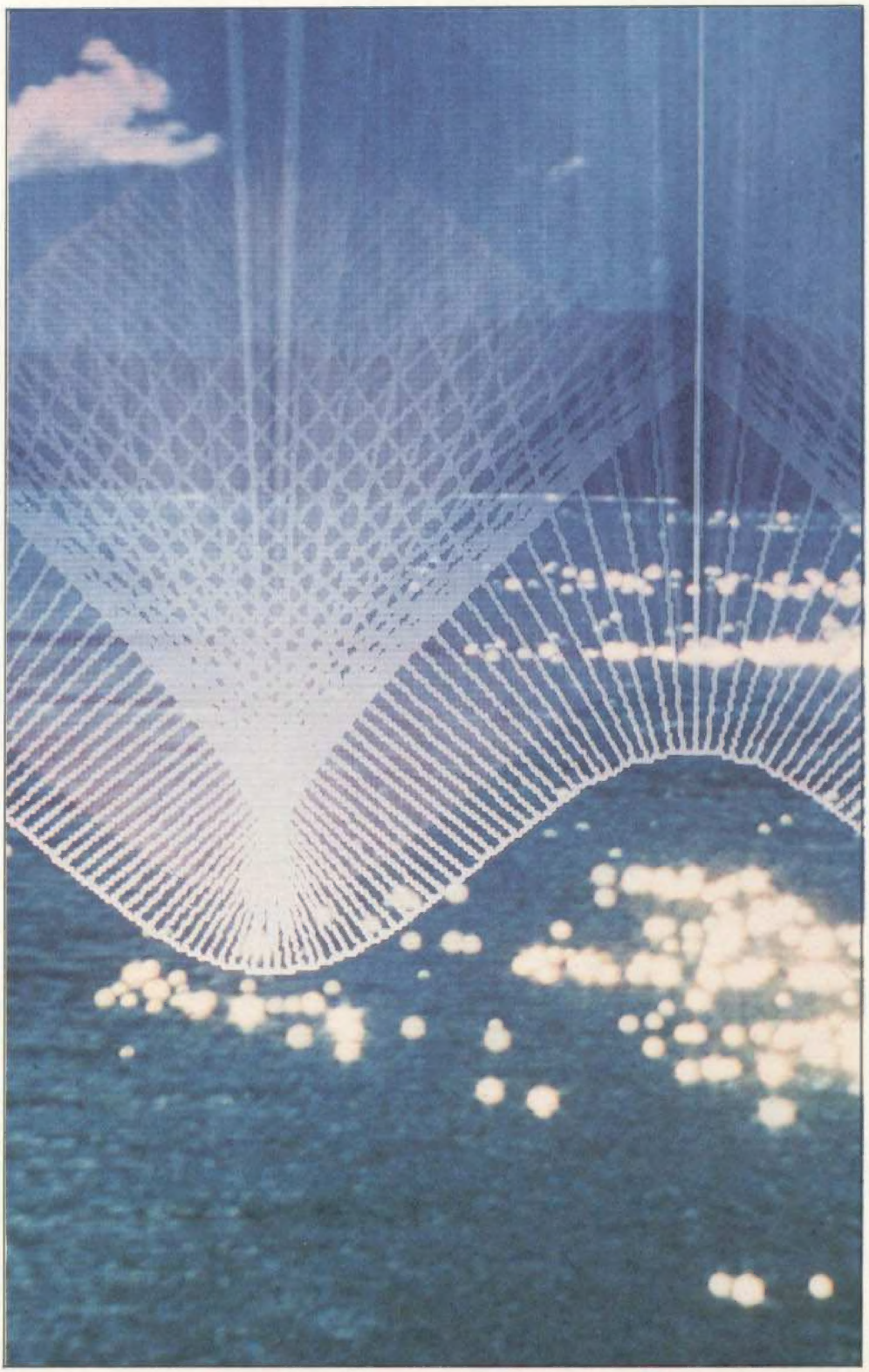
Preparador de Texto/Coordenador: Eliel Silveira Cunha
Preparadores de Texto: Alzira Moreira Braz, Ana Maria Dilguerian, Levon Yacubian, Luciano Tasca, Maria Teresa Galluzzi, Maria Teresa Martins Lopes, Paulo Felipe Mendrone
Revisor/Coordenador: José Maria de Assis
Revisoras: Conceição Aparecida Gabriel, Isabel Leite de Camargo, Ligia Aparecida Ricetto, Maria de Fátima Cardoso, Nair Lucia de Britto
Paste-up: Anastase Potaris, Balduino F. Leite, Edson Donato

© Marshall Cavendish Limited 1984/85.
© Editora Nova Cultural Ltda., São Paulo, Brasil, 1986.
Edição organizada pela Editora Nova Cultural Ltda.
Av. Brigadeiro Faria Lima, n.º 2000 - 3.º andar
CEP 01452 - São Paulo - SP - Brasil
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta na AM Produções Gráficas Ltda. e impressa na Divisão Gráfica da Editora Abril S.A.

PADRÕES NATURAIS

- FAMÍLIAS DE CURVAS
- ENVOLTÓRIAS
- FOCOS E CÚSPIDES
- PADRÕES DE PONTOS
- DINÂMICA DE POPULAÇÕES

Com rotinas gráficas muito simples, podemos criar uma variedade surpreendente de desenhos — todos eles estruturados a partir de padrões fornecidos pela natureza.



A utilização do computador para desenhar a trajetória de objetos em queda livre sob a ação da gravidade foi examinada nos artigos publicados nas páginas 766 e 781. Com os programas apresentados, ilustramos alguns aspectos de um velho ramo da Física chamado *dinâmica* e vimos como traçar parábolas, círculos e elipses. Neste artigo, aprenderemos a desenhar outros tipos de curva usando técnicas semelhantes. Rotinas gráficas bastante simples se encarregarão dessa tarefa.

Nos últimos anos, a dinâmica recebeu um novo impulso enquanto campo de pesquisa. Um dos motivos para isso foi o reconhecimento de que seus princípios matemáticos não se aplicam somente ao movimento dos corpos sob a ação de determinadas forças. Cientistas ópticos, interessados na refração da luz das estrelas na atmosfera, químicos industriais que estudam o desencadeamento de reações e biólogos preocupados com o crescimento de populações animais que competem entre si — todos têm recorrido aos conhecimentos acumulados por esse ramo da Física. Além disso, os computadores — permitindo a repetição rápida de operações elementares — favoreceram a revelação de funções de uma complexidade estrutural que os antigos métodos de cálculo sequer sugeriam. Com freqüência, um padrão ou curva tem que ser repetido inúmeras vezes antes que possamos notar a presença de uma estrutura subjacente. Os programas abaixo demonstram isso.

FAMÍLIAS DE CURVAS

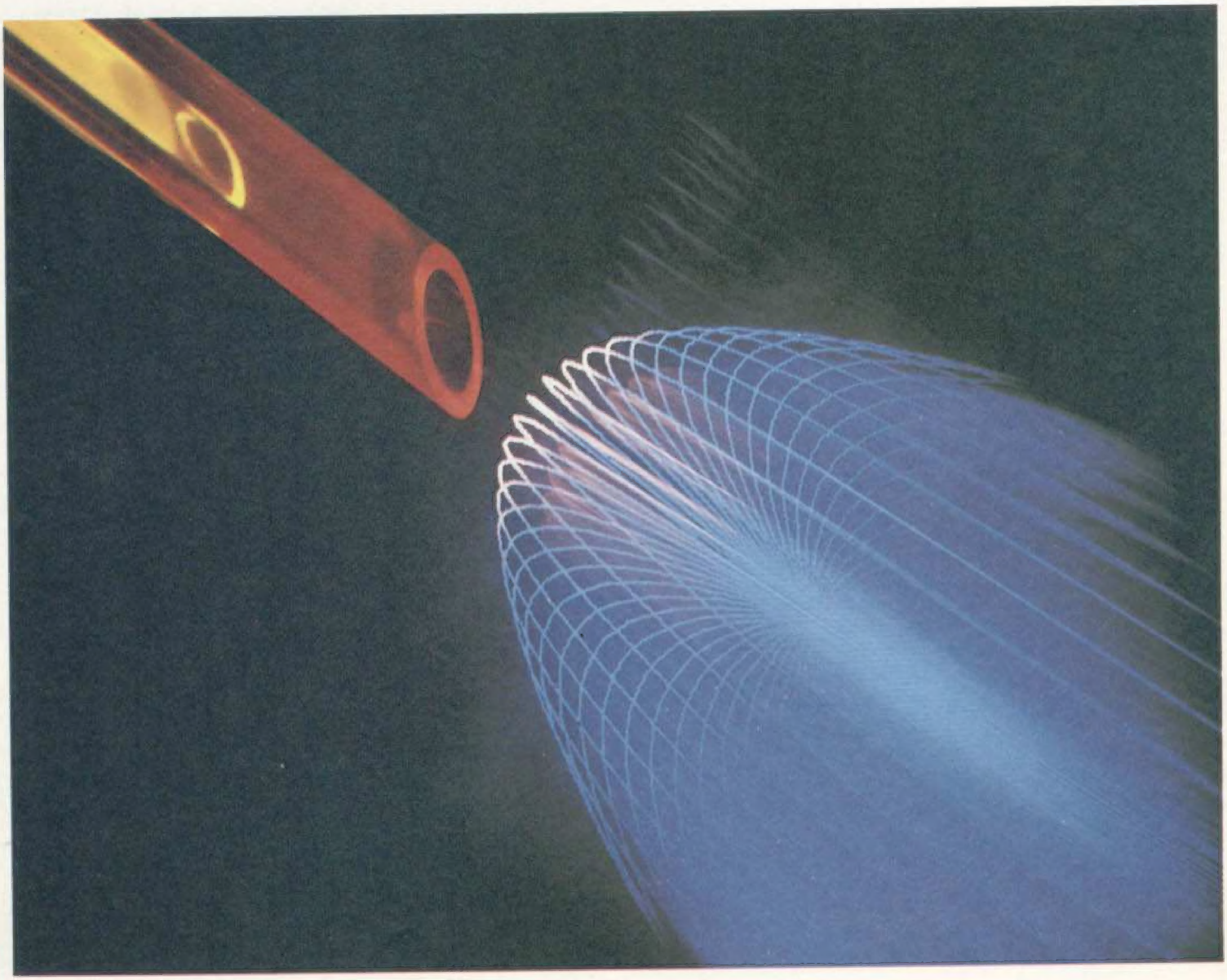
Algumas curvas, isoladamente simples, criam padrões bastante complexos quando são reunidas, constituindo uma família. Veja um exemplo em que se utilizam parábolas.

```
 5 LET A$="": FOR N=1 TO 64:
 LET A$=A$+" ": NEXT N
10 BRIGHT 1: BORDER 0: INK 5:
 PAPER 0: CLS
20 LET NMAX=81
30 LET DELT=.05
40 LET SX=170/SQR 3: LET SY=
175
50 FOR N=1 TO NMAX
60 LET A=PI*(-1+2*N/NMAX)
70 PLOT 128,80
80 FOR T=0 TO 3 STEP DELT
90 LET X=T*COS A: LET Y=T*(
SIN A-T/2)
100 IF Y<=-.4 THEN  GOTO 120
110 LET DX=SX*X+128: LET DY=SY
*Y+80
111 IF DX<0 OR DX>255 OR DY<0
OR DY>175 THEN  GOTO 120
115 DRAW DX-PEEK 23677,DY-PEEK
23678
117 PRINT AT 19,0;A$
120 NEXT T
130 NEXT N
```

```
10 COLOR 15,1,1:SCREEN 2
20 NM=81:PI=4*ATN(1)
30 DE=.05
40 SX=160/SQR(3):SY=230
50 FOR N=1 TO NM
60 A=PI*(-1+2*N/NM)
70 DRAW"BM127,118"
80 FOR T=0 TO 3 STEP DE
90 X=T*COS(A):Y=T*(SIN(A)-T/2)
100 IF Y>-.4 THEN LINE-(SX*X+12
7,118-SY*Y) ELSE T=3
110 NEXT T
120 NEXT N
130 GOTO 130
```

```
10  HGR2
20 NM = 81:PI = 4 *  ATN (1)
30 DE = .05
40 SX = 160 /  SQR (3):SY = 180
50  FOR N = 1 TO NM
60 A = PI * ( - 1 + 2 * N / NM)
70  HCOLOR= 3:X1 = 139:Y1 = 118
80  FOR T = 0 TO 3 STEP DE
90 X = T *  COS (A):Y = T * ( S
IN (A) - T / 2)
100  IF Y >  - .35 THEN  HPLOT
X1,Y1 TO SX * X + 139,118 - SY
* Y:X1 = SX * X + 139:Y1 = 118
- SY * Y: GOTO 110
105 T = 3
110  NEXT T
120  NEXT N
```

```
10 PMODE 4,1:PCLS 1:SCREEN 1,0
20 NM=81:PI=4*ATN(1)
30 DE=.05
40 SX=160/SQR(3):SY=230
50 FOR N=1 TO NM
60 A=PI*(-1+2*N/NM)
70 DRAW"BM127,118"
80 FOR T=0 TO 3 STEP DE
90 X=T*COS(A):Y=T*(SIN(A)-T/2)
100 IF Y>-.4 THEN LINE-(SX*X+12
```



```
7,118-SY*Y),PRESET ELSE T=3
110 NEXT T
120 NEXT N
130 GOTO 130
```

O programa simula as trajetórias de jatos de água lançados por um esguicho de jardim, ou, em uma interpretação mais moderna, de nêutrons lançados de um tubo fino ligado a um reator. Em ambos os casos, a família de curvas é constituída por todas as trajetórias parabólicas que emergem no mesmo ponto, em direções variadas mas com a mesma velocidade. O padrão formado por essa família é a curva exterior que todas as curvas tocam. A curva exterior, chamada de *envoltória*, é também, neste caso, uma parábola. Em balística, usa-se o termo parábola limitante para uma curva desse tipo, pois ela define os limites da região que pode ser alcançada por projéteis que tenham uma determinada velocidade inicial.

É importante notar que a envoltória é uma propriedade da família de curvas como um todo, não tendo nenhum significado para uma trajetória em particular. Trata-se de um bom exemplo de que o todo pode ser mais do que a simples soma de suas partes.

No programa, a equação das curvas aparece na linha 90. **X** é a distância horizontal; **Y**, a distância vertical; **T** corresponde ao tempo (começando por zero no instante do lançamento) e cada trajetória tem uma direção determinada por **A**, o ângulo ou inclinação do lançamento. O programa desenha **NMAX** ou **NM** inclinações; tente mudar o valor dessa variável na linha 20.

FOCOS E CÚSPIDES

Este programa mostra como também as linhas retas podem constituir famílias com envoltórias interessantes.

```
10 BORDER 0: INK 7: PAPER 0:
CLS : LET N=2
30 LET Y0=80/N/N
40 PLOT 0,-6-Y0*2
50 FOR X=0 TO 255 STEP 2
60 LET Y=169-Y0-Y0*COS (N*2*
PI*X/255)
70 DRAW X-PEEK 23677,175-Y-
PEEK 23678
80 LET XT=X+2*Y0*N*PI*Y/255*
SIN (N*2*PI*X/255)
85 LET Y1=0: IF XT<0 THEN
LET XT=0: LET Y1=Y+255*X/(2*N
*PI*Y0*SIN (N*2*PI*X/255))
86 IF XT>255 THEN  LET XT=255
: LET Y1=Y-(255-X)*255/(2*N*
PI*Y0*SIN (N*2*PI*X/255))
90 DRAW XT-PEEK 23677,175-Y1-
PEEK 23678
100 PLOT X,175-Y
110 NEXT X
120 GOTO 120
```

```
10 COLOR 15,1,1:SCREEN 2
20 N=2:PI=4*ATN(1)
30 Y0=80/N/N
40 DRAW"BM0,"+STR$(INT(186-Y0*2
))
50 FOR X=0 TO 255 STEP 2
60 Y=186-Y0-Y0*COS(N*2*PI*X/255
)
70 LINE -(X,Y)
80 XT=X+2*Y0*N*PI*Y/255*SIN(N*2
*PI*X/255)
85 Y1=0:IF XT<0 THEN XT=0:Y1=Y+
255*X/(2*N*PI*Y0*SIN(N*2*PI*X/2
55))
86 IF XT>255 THEN XT=255:Y1=Y-(
255-X)*255/(2*N*PI*Y0*SIN(N*2*P
I*X/255))
90 LINE -(XT,Y1)
100 DRAW"BM"+STR$(X)+","+STR$(I
NT(Y))
110 NEXT
120 GOTO 120
```

```
10  HGR2
20 N = 2:PI = 4 *  ATN (1)
30 Y0 = 80 / N / N
40  HCOLOR= 3:XX = 0:YY = 186 -
 Y0 * 2
50  FOR X = 0 TO 279 STEP 2
60 Y = 186 - Y0 - Y0 *  COS (N
* 2 * PI * X / 279)
70  HPLOT XX,YY TO X,Y
80 XT = X + 2 * Y0 * N * PI * Y
 / 279 *  SIN (N * 2 * PI * X /
 279)
85 Y1 = 0: IF XT < 0 THEN XT =
0:Y1 = Y + 279 * X / (2 * N * P
I * Y0 *  SIN (N * 2 * PI * X /
 279))
86  IF XT > 279 THEN XT = 279:Y
1 = Y - (279 - X) * 279 / (2 *
N * PI * Y0 *  SIN (N * 2 * PI
* X / 279))
90  HPLOT X,Y TO XT,Y1
100 XX = X:YY = Y
110  NEXT
```

```
10 PMODE 4,1:PCLS1:SCREEN 1,0
20 N=2:PI=4*ATN(1)
30 Y0=80/N/N
40 DRAW"BM0,"+STR$(INT(186-Y0*2
))
50 FOR X=0 TO 255 STEP 2
60 Y=186-Y0-Y0*COS(N*2*PI*X/255
)
70 LINE -(X,Y),PSET
80 XT=X+2*Y0*N*PI*Y/255*SIN(N*2
*PI*X/255)
85 Y1=0:IF XT<0 THEN XT=0:YZ=Y+
255*X/(2*N*PI*Y0*SIN(N*2*PI*X/2
55))
86 IF XT>255 THEN XT=255:Y1=Y-(
255-X)*255/(2*N*PI*Y0*SIN(N*2*P
I*X/255))
90 LINE -(XT,YQ),PRESET
100 DRAW"BM"+STR$(X)+","+STR$(I
NT(Y))
110 NEXT
120 GOTO 120
```

O programa simula o reflexo de raios de luz sobre uma superfície ondulada — como o sol batendo nas águas de uma lagoa. Esta família é composta somente por linhas retas, que formam ângulos retos com uma curva senóide. A envoltória consiste na curva constituída pelos pontos focais. Estes, no caso dos raios, são mais intensos nos vales da senóide, onde a envoltória tem pontos brilhantes chamados *cúspides*. A existência de cúspides pode ser prevista através de um novo ramo da Matemática denominado *Teoria das Catástrofes* (esse nome dramático vem da aplicação da teoria ao desabamento de pontes e emborcação de navios).

No programa, a senóide é especificada na linha 60 e os raios de luz, na 80. O número de vales da senóide é **N** (menos no Apple). Tente mudar o valor de **N** na linha 20 (recomendamos **N** = 1).

Senóides também podem constituir uma família de curvas. Veja o programa apresentado a seguir.

```
10 BORDER 0: PAPER 0: INK 7:
CLS
20 LET QM=SQR (2*LN (3))
30 FOR Q=-QM TO QM*1.001 STEP
QM/30
40 PLOT 0,75+Q*75/QM
50 FOR T=0 TO 3*PI STEP .2
60 LET X=Q*COS (EXP (-Q*Q/2)*
T)
70 DRAW (T*240/3/PI)-PEEK
23677,75+(X*75/QM)-PEEK 23678
80 NEXT T
90 NEXT Q
```

```
10 COLOR 15,1,1:SCREEN 2
20 QM=SQR(2*LOG(3)):PI=4*ATN(1)
30 FOR Q=-QM TO QM*1.001 STEP Q
M/30
40 DRAW "BM0,"+STR$(INT(100-Q*9
0/QM))
50 FOR T=0 TO 3*PI STEP .2
60 X=Q*COS(EXP(-Q*Q/2)*T)
70 LINE-(T*240/3/PI,100-X*90/QM
)
80 NEXT T
90 NEXT Q
100 GOTO 100
```

```
10  HGR2
20 QM =  SQR (2 *  LOG (3)):PI
= 4 *  ATN (4)
```

```
30  FOR Q =  - QM TO QM * 1.001
 STEP QM / 30
40  HCOLOR= 3:X1 = 0:Y1 = 100 -
 Q * 90 / QM
50  FOR T = 0 TO 3.5 * PI STEP
.2
60 X = Q *  COS ( EXP ( - Q * Q
 / 2) * T)
70  HPLOT X1,Y1 TO T * 240 / 3
/ PI,100 - X * 90 / QM:X1 = T *
 240 / 3 / PI:Y1 = 100 - X * 90
 / QM
80  NEXT T
90  NEXT Q
```

```
10 PMODE 4,1:PCLS 1:SCREEN 1,0
20 QM=SQR(2*LOG(3)):PI=4*ATN(1)
30 FOR Q=-QM TO QM*1.001 STEP Q
M/30
40 DRAW"BM0,"+STR$(INT(100-Q*90
/QM))
50 FOR T=0 TO 3*PI STEP .2
60 X=Q*COS(EXP(-Q*Q/2)*T)
70 LINE -(T*240/3/PI,100-X*90/Q
M),PRESET
80 NEXT T
90 NEXT Q
100 GOTO 100
```

Simulamos aqui raios de luz atravessando uma fibra óptica cujo índice de refração varia ao longo de sua largura. Em uma escala microscópica, poderíamos estar representando também "trajetórias" de elétrons entre dois planos de átomos de um cristal colocado sob o canhão de um microscópio eletrônico. A família é composta por senóides que começam no canto esquerdo do vídeo, paralelas umas às outras; o período de cada curva depende de sua posição inicial. Embora esta família seja bem diferente da anterior, a envoltória também tem focos e cúspides.

No programa, as órbitas são especificadas na linha 60, que contém a distância **X** do eixo principal no tempo **T**. As senóides são definidas por **Q**. Como está, o programa desenha trinta senóides; para obter um número maior ou menor, modifique a linha 30.

PADRÕES DE PONTOS

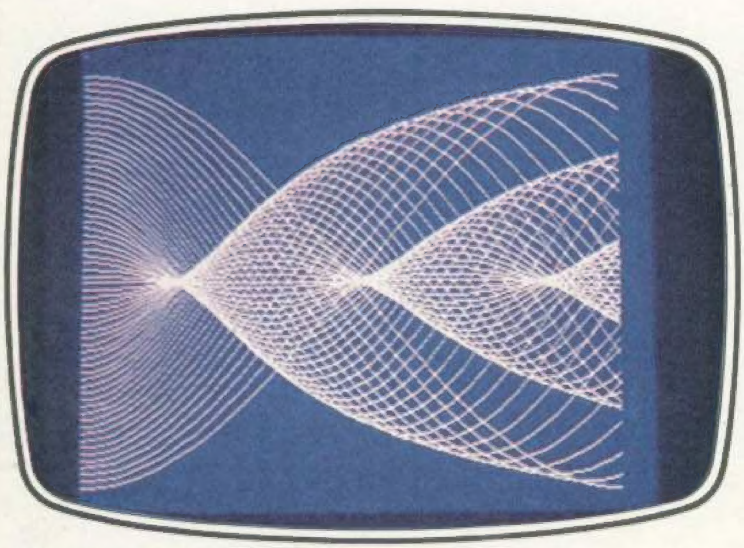
Juntar curvas em famílias não é a única maneira de obter padrões interessantes. Um outro caminho é seguir uma órbita por um longo tempo, de modo que uma estrutura complexa possa se revelar, ainda que a fórmula matemática da curva seja relativamente simples. Para desenhar esse tipo de curva, não devemos fazer um traçado contínuo, pois, após alguns segundos, teremos preenchido a tela com um emaranhado de curvas. O mais conveniente é traçar a órbita por meio de pontos, dispostos na tela a intervalos regulares — como se a trajetória fosse observada sob luz estroboscópica. Os três programas seguintes exemplificam essa técnica. A posição dos pontos na tela nem sempre equivale à posição real do objeto de estudo; o que o programa fornece, muitas vezes, é uma representação abstrata, na qual a coordenada horizontal corresponde à posição real e a coordenada vertical, à velocidade.

Este programa leva vários minutos para completar o desenho:

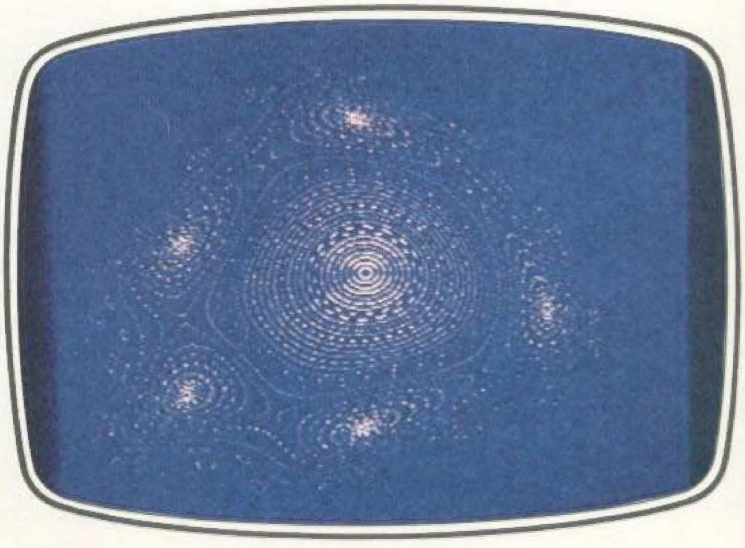
```
10 BORDER 0: PAPER 0: INK 7:
CLS
30 LET A=76.11
40 LET ALF=A*PI/180: LET C=
COS (ALF)
50 LET S=SIN (ALF)
 60 LET NMAX=200
 70 LET M=52
 80 FOR J=1 TO M
 90 LET X=0: LET Y=J/M
100 FOR N=1 TO NMAX
110 LET W=X
120 LET X=X*C-(Y-X*X)*S: LET Y
=W*S+(Y-W*W)*C
130 IF ABS (X)>4 OR ABS (Y)>4
THEN  GOTO 860
135 IF X>1 OR Y>1 THEN  GOTO
 150
140 PLOT X*128+128,Y*85+85
150 NEXT N
160 NEXT J
```

```
10 COLOR 15,1,1:SCREEN 2
20 A=76.11:PI=4*ATN(1)
30 AL=A*PI/180:C=COS(AL)
40 S=SIN(AL)
50 NM=200
60 M=52
70 FOR J=1 TO M
80 X=0:Y=J/M
90 FOR N=1 TO NM
100 W=X
110 X=X*C-(Y-X*X)*S:Y=W*S+(Y-W*
W)*C
120 IF ABS(X)>4 OR ABS(Y)>4 THE
N 160
125 IF ABS(Y)>1 OR ABS(X)>1 THE
N 140
130 PSET(128+X*128,96-Y*96)
140 NEXT N
150 NEXT J
160 GOTO 160
```

```
10  HGR2
20 A = 76.11:PI = 4 *  ATN (1)
30 AL = A * PI / 180:C =  COS (
AL)
40 S =  SIN (AL)
50 NM = 200
60 M = 52
70  FOR J = 1 TO M
```



Raios de luz em uma fibra óptica.



"Ilhas de pontos" ou linhas de força.

```
80 X = 0:Y = J / M
90  FOR N = 1 TO NM
100 W = X
110 X = X * C - (Y - X * X) * S
:Y = W * S + (Y - W * W) * C
120  IF  ABS (X) > 4 OR  ABS (Y
) > 4 THEN 160
125  IF  ABS (X) > 1 OR  ABS (Y
) > 1 THEN 140
130  HPLOT 128 + X * 128,96 - Y
 * 96
140  NEXT N
150  NEXT J
160  GOTO 160
```

```
10 PMODE 4,1:PCLS1:SCREEN 1,0
20 A=76.11:PI=4*ATN(1)
30 AL=A*PI/180:C=COS(AL)
40 S=SIN(AL)
50 NM=200
60 M=52
70 FOR J=1 TO M
80 X=0:Y=J/M
90 FOR N=1 TO NM
100 W=X
110 X=X*C-(Y-X*X)*S:Y=W*S+(Y-W*
W)*C
120 IF ABS(X)>4 OR ABS(Y)>4 THE
N 160
125 IF ABS(Y)>1 OR ABS(X)>1 THE
N 140
130 PRESET(128+X*128,96-Y*96)
140 NEXT N
150 NEXT J
160 GOTO 160
```

O programa simula o movimento de partículas subatômicas (como prótons e elétrons) em um acelerador ou, ainda, linhas de força de um campo magnético. Sua atuação, em termos matemáticos, consiste em tomar uma série de pontos iniciais e movê-los pela tela, seguindo esta regra: "faça uma rotação em torno do centro da tela, segundo um ângulo fixo, só que com uma ligeira modificação". Sem esta "ligeira modificação", todas as curvas seriam círculos concêntricos — e, de fato, as mais próximas do centro se assemelham a círculos. O efeito dessa modificação tem maior impacto sobre as curvas externas, que acabam formando um conjunto de "ilhas". Se pudéssemos ampliar mais o desenho, distinguiríamos várias ilhas menores, distribuídas entre as que podemos observar agora.

No programa, o número de pontos iniciais é **M**, na linha 60: caso você não queira esperar tanto para ver o desenho completo, diminua esse valor. Cada ponto é desenhado **NMAX** ou **NM** vezes; o valor é especificado na linha 50. A regra fica nas linhas 100 e 110, que determinam de que modo as coordenadas vertical e horizontal serão modificadas a cada repetição. O ângulo fixo de rotação é dado por **A** em graus, na linha 20; experimente valores diferentes (recomendamos 90).

O programa apresentado a seguir gera um padrão surpreendente, a partir de um ponto inicial:

```
20 LET K=51.3: BORDER 0: INK
7: PAPER 0: CLS
30 LET X=1/PI: LET P=0
40 LET A=1/SQR 5
50 FOR N=1 TO 10000
60 LET Y=X-.5: LET X=X+A-INT
(X+A)
70 LET P=P-Y
80 PLOT X*255,P*K+85
90 NEXT N
```

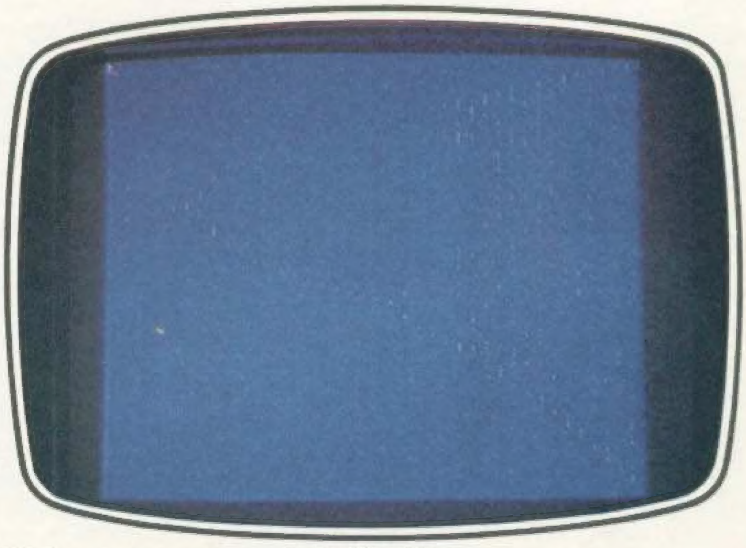
```
10 COLOR 15,1,1:SCREEN 2
20 K=60:X=1/(4*ATN(1)):P=0
30 A=1/SQR(5)
40 FOR N=1 TO 10000
50 Y=X-.5:X=X+A-INT(X+A)
60 P=P-Y
70 PSET(X*255,128-P*K)
80 NEXT N
90 GOTO 90
```

```
10  HGR2 :K = 40
20 X = 1 / (4 *  ATN (1)):P = 0
30 A = 1 /  SQR (5)
40  FOR N = 1 TO 10000
50 Y = X - .5:X = X + A -  INT
(X + A)
60 P = P - Y
70  HCOLOR= 3: HPLOT X * 279,10
0 - P * K
80  NEXT N
```

```
10 PMODE 4,1:PCLS1:SCREEN 1,0:K
=60
20 X=1/(4*ATN(1)):P=0
30 A=1/SQR(5)
40 FOR N=1 TO 10000
50 Y=X-.5:X=X+A-INT(X+A)
60 P=P-Y
70 PRESET (X*255,128-P*K)
80 NEXT N
90 GOTO 90
```

Esse programa é uma simulação da *ressonância*, que ocorre quando dois fenômenos físicos têm freqüências coincidentes ou múltiplas. Por exemplo, um dos fenômenos pode ser a órbita de um asteróide em torno do sol; o segundo, a perturbação dessa órbita pela atração gravitacional de Júpiter. Podemos nos perguntar: as perturbações causadas pelo grande planeta serão capazes de arrancar o asteróide de sua órbita? Tudo depende da *razão* entre as duas freqüências (a razão é obtida dividindo-se uma pela outra). As partículas dos anéis de

Uma regra simples produz o caos.



Mudanças em uma população de peixes.

Saturno estão sujeitas a esse tipo de efeito.

No programa, a razão é chamada de **A** e seu valor (menor que 1) está na linha 30. O número de repetições da ressonância é 10000 (final da linha 40); reduza esse valor caso não queira esperar muito. A transformação da ressonância, nas linhas 50 e 60, é obtida por meio de uma regra que modifica as posições horizontal (**X**) e vertical (**P**) dos pontos na tela.

Um delicado padrão em "cortina" é obtido com **A = 1/SQR(5)**, ou seja, 0.4472136. Como esse valor não corresponde à razão entre dois números inteiros (em termos do computador, trata-se de um número irracional), ele é não-ressonante. Para chegar à ressonância, temos que atribuir a **A** um valor que seja a razão entre dois números inteiros — como 9/20, que vale 0.45. Tente também um outro número irracional, como **1/PI**. Se quiser reduzir a escala vertical, diminua o valor de **K**.

CAOS

Observe que certas regras simples não produzem nenhum padrão. O próximo programa ilustra o fenômeno.

```
10 BORDER 0: PAPER 0: INK 7:
CLS
20 LET X=1/PI
30 LET Y=1/PI
40 FOR M=1 TO 10000
50 LET X=X+Y-INT (X+Y)
60 LET Y=X+Y-INT (X+Y)
70 PLOT X*255,Y*175
80 NEXT M
```

```
10 COLOR 15,1,1:SCREEN 2
20 PI=4*ATN(1):X=1/PI
30 Y=1/PI
40 FOR M=1 TO 10000
50 X=X+Y-INT(X+Y)
60 Y=X+Y-INT(X+Y)
70 PSET(X*255,192-Y*191)
80 NEXT M
90 GOTO 90
```

```
10  HGR2
20 PI = 4 *  ATN (1):X = 1 / PI
30 Y = 1 / PI
40  FOR M = 1 TO 10000
50 X = X + Y -  INT (X + Y)
60 Y = X + Y -  INT (X + Y)
70  HCOLOR= 3: HPLOT X * 279,19
2 - Y * 191
80  NEXT M
```

```
10 PMODE 4,1:PCLS1:SCREEN 1,0
20 PI=4*ATN(1):X=1/PI
30 Y=1/PI
40 FOR M=1 TO 10000
50 X=X+Y-INT(X+Y)
60 Y=X+Y-1NT(X+Y)
70 PRESET(X*255,192-Y*191)
80 NEXT M
90 GOTO 90
```

Esse programa simula o vaivém errático das bolas de um fliperama, o movimento das moléculas de um gás, ou qualquer outro sistema de órbita tão imprevisível que se torna impossível distingui-la de um processo aleatório. Convém notar, porém, que os 10.000 pontos desenhados na tela não são aleatórios, resultando da repetição de uma regra determinística, ou seja, que não abriga nenhum elemento de acaso. A regra considera a tela como um quadrado unitário onde as coordenadas verticais e horizontais variam entre zero e um. Ela opera em três etapas, nas linhas 50 e 60 do programa. Primeiro, as coordenadas **X** e **Y** do último ponto são somadas para produzir um novo **X**; depois, esse **X** é somado a **Y**, resultando em outro **Y**; finalmente, se **X** ou **Y** forem maiores que 1, um número inteiro apropriado é subtraído para que as coordenadas permaneçam no intervalo válido.

POPULAÇÕES DE PEIXES

Como a população de peixes de um lago varia ao longo das gerações? Isto depende dos fatores que condicionam as mudanças populacionais de uma geração para a outra. Para definir uma regra, devemos considerar tanto a tendência ao crescimento populacional resultante da reprodução, como a tendência à redução, determinada pela quantidade de alimento disponível no lago. Conforme o equilíbrio estabelecido entre esses dois fatores, a população pode ficar estável, alternar entre um ou mais valores, ou variar de tamanho ao acaso, ao longo das gerações.

O programa dado a seguir ilustra o fenômeno. A posição horizontal, **A**, corresponde à razão entre disponibilidade de alimento e taxa de natalidade, representando assim as sucessivas gerações. A posição vertical **Y** corresponde ao tamanho da população, que aumenta ou diminui alguns pontos conforme as gerações se sucedem. O tamanho da população **Y** só é colocado na tela quando se encontra em equilíbrio, isto é, quando assume um valor estável ou alterna entre dois ou mais valores, que são denominados *pontos de atração*. No Spectrum e no TRS-Color, um som de freqüência proporcional ao tamanho da população é emitido.

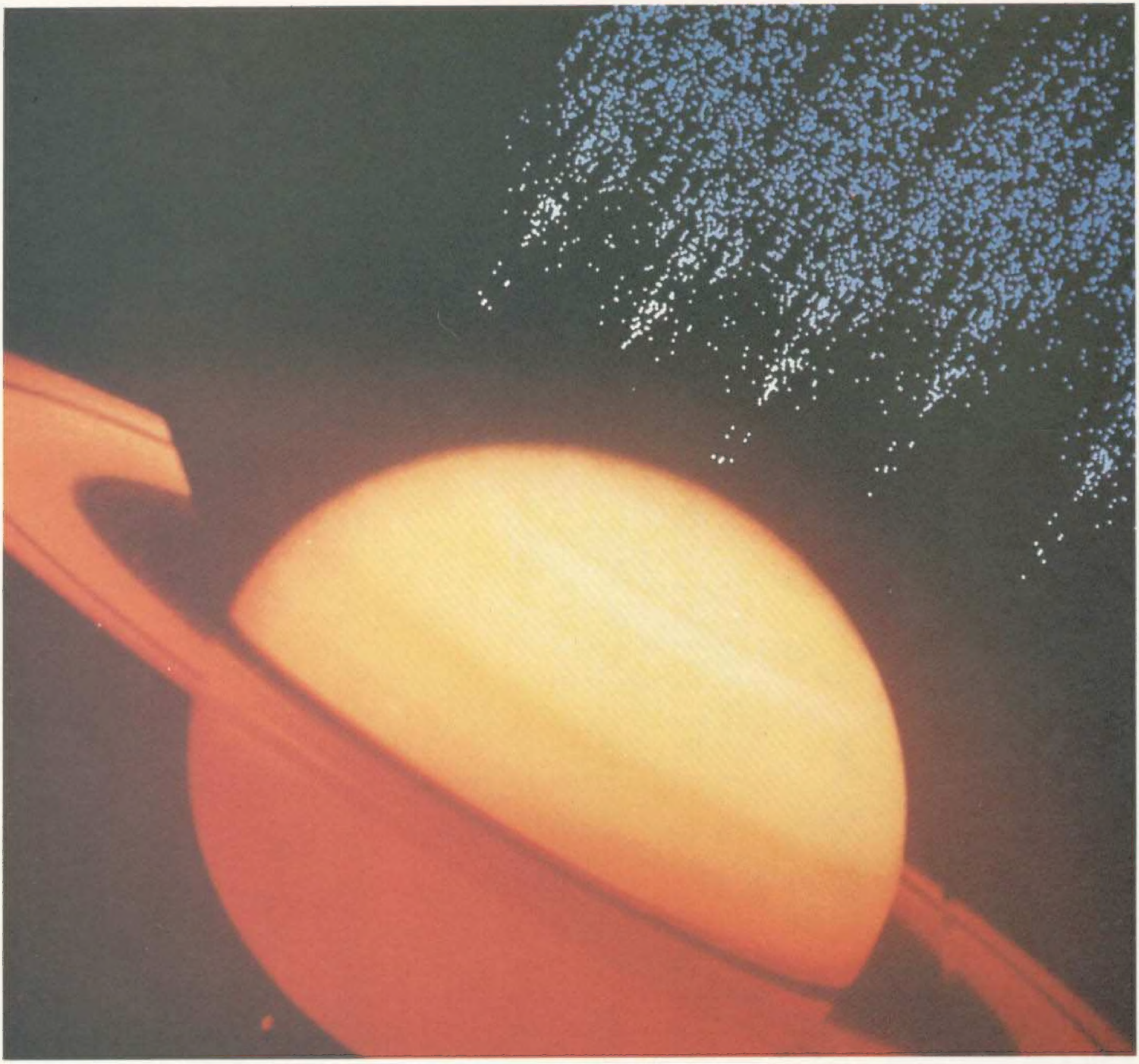
```
10 BORDER 0: INK 7: PAPER 0:
CLS
20 LET S=.03: LET NMIN=50:
LET NMAX=80
30 FOR A=2.8 TO 4 STEP S
40 LET Y=1/PI
50 FOR N=1 TO NMAX
60 LET Y=A*Y*(1-Y)
70 IF N>NMIN THEN  PLOT 255*(
A-2.8)/1.2,Y*175
80 SOUND .0075,Y*20
90 NEXT N
100 NEXT A
```

```
10 COLOR 15,1,1:SCREEN 2
20 S=1/127:NN=50:NX=80
30 FOR A=2.8 TO 4 STEP S
40 Y=.25/ATN(1)
50 FOR N=1 TO NX
60 Y=A*Y*(1-Y)
70 IF N>NN THEN PSET(255*(A-2.8
)/1.2,192-Y*191)
90 NEXT N
100 NEXT A
110 GOTO 110
```

```
10  HGR2
20 S = 1 / 127:NN = 50:NX = 80
30  FOR A = 2.8 TO 4 STEP S
40 Y = .25 /  ATN (1)
50  FOR N = 1 TO NX
60 Y = A * Y * (1 - Y)
70  IF N > NN THEN  HPLOT 279 *
 (A - 2.8) / 1.2,192 - Y * 192
90  NEXT N
100  NEXT A
```

```
10 PMODE 4,1:PCLS1:SCREEN 1,0
20 S=1/127:NN=50:NX=80
30 FOR A=2.8 TO 4 STEP S
40 Y=.25/ATN(1)
50 FOR N=1 TO NX
60 Y=A*Y*(1-Y)
70 IF N>NN THEN PRESET (255*(A-
2.8)/1.2,192-Y*191)
80 SOUND 1+255*Y,1
90 NEXT N
100 NEXT A
110 GOTO 110
```

Como você pode observar, os pontos de atração modificam-se drasticamente à medida que a disponibilidade de alimento aumenta. No início (lado esquerdo da tela), só há um valor de atração, indicando uma população estável. Isso significa que o alimento é suficiente para os peixes sobreviverem e se reproduzirem. Se a população crescer demais,



haverá menos alimento e alguns peixes morrerão. Com a maior disponibilidade de alimento para os sobreviventes, a população volta a crescer, podendo, em seguida, passar por uma nova fase de estabilidade.

Subitamente, quando **A** assume um certo valor, a quantidade de alimento ultrapassa determinado limite. Passam a existir, então, dois pontos de atração e a curva se bifurca — indicando que o tamanho da população agora se alterna entre dois valores. Mais tarde, quando a quantidade de alimento se torna ainda maior, a curva se bifurca mais uma vez, aumentando o número de pontos de atração. Mais e mais divisões vão ocorrendo em uma seqüência infinita, da qual o programa só tem resolução para mostrar os primeiros passos. As bifurcações se acumulam e infinitos pontos de atração passam a corresponder a um certo valor finito de **A**. Em conseqüência, a população varia sem nunca se estabilizar.

A descoberta dessa "árvore de bifurcações", que acaba num completo caos, despertou grande interesse na comunidade científica, por ter diversas aplicações. Ela retrata, por exemplo, a modificação do fluxo de um líquido, que passa de um estágio moderado (ponto de atração estável) para um turbulento (caos), conforme a velocidade aumenta. O mesmo efeito pode ser observado na fumaça de um cigarro, que sobe suavemente e de repente desenvolve espirais, terminando em turbulência.

No programa, a lei de evolução está na linha 60. Para ampliar a resolução, altere a linha 20 de forma que **S** = .005, **NMIM** ou **NM** = 200 e **NMAX** ou **NX** = 300. No Spectrum e no TRS-Color, apague a linha 80 para aumentar a velocidade do programa.

AVALANCHE: OS SALTOS DE WILLIE

Nosso personagem não pode mesmo confiar em sua sorte. Surpreendido por uma avalanche tão logo começou a caminhada, conta apenas com um recurso para escapar das pedras: saltar sobre elas.

A parte do programa aqui apresentada permite que Willie se movimente para cima e para baixo, no mesmo lugar — ou seja, nosso pobre personagem aprende, a tempo, a saltar sobre as pedras que se aproximam.

Willie já pode andar, mas isso não o ajudará muito na hora de enfrentar a avalanche. Vamos fazê-lo saltar, de modo que as pedras passem rolando sob suas pernas, sem atingi-lo.

```
 10 REM org 59336
 20 REM jmp ld de,6
 30 REM ld hl,759
 40 REM call 949
 50 REM ld a,(57335)
 60 REM cpl
 70 REM jr nz,mjb
 80 REM inc a
 90 REM ld (57335),a
100 REM ld hl,(57332)
110 REM ld de,32
120 REM sbc hl,de
130 REM ld (57332),hl
140 REM ld bc,57048
150 REM ld a,40
160 REM ld de,259
170 REM call 58970
180 REM ret
190 REM mjb cp 2
200 REM jr nz,mjc
210 REM inc a
220 REM ld (57335),a
230 REM ld hl,(57332)
240 REM ld bc,57000
250 REM ld a,40
260 REM ld de,258
270 REM call 58970
280 REM ld de,64
290 REM add hl,de
300 REM ld a,45
310 REM ld bc,15616
320 REM call 58217
330 REM ret
340 REM mjc cp 3
350 REM jr nz,mjd
360 REM inc a
370 REM ld (57335),a
380 REM ld hl,(57332)
390 REM ld bc,57048
400 REM ld a,40
410 REM ld de,259
420 REM call 58970
430 REM ld de,32
440 REM add hl,de
450 REM ld (57332),hl
460 REM ret
470 REM mjd cp 4
480 REM jr nz,mfj
490 REM ld a,0
500 REM ld (57335),a
510 REM ld hl,(57332)
520 REM ld de,32
530 REM sbc hl,de
540 REM ld bc,15616
550 REM ld a,45
560 REM call 58217
570 REM jp 59153
580 REM mfl ret
```

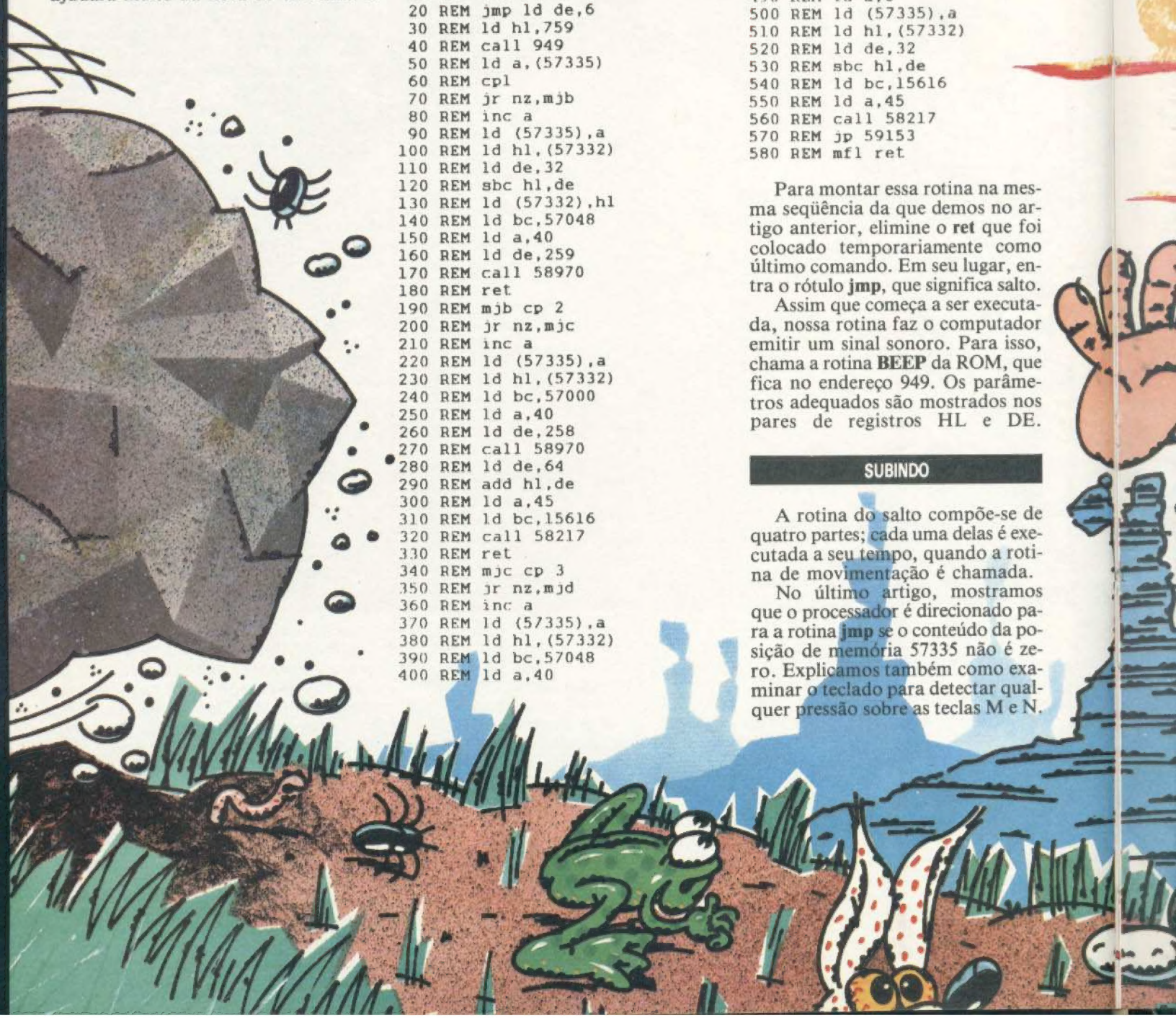
Para montar essa rotina na mesma seqüência da que demos no artigo anterior, elimine o **ret** que foi colocado temporariamente como último comando. Em seu lugar, entra o rótulo **jmp**, que significa salto.

Assim que começa a ser executada, nossa rotina faz o computador emitir um sinal sonoro. Para isso, chama a rotina **BEEP** da ROM, que fica no endereço 949. Os parâmetros adequados são mostrados nos pares de registros HL e DE.

SUBINDO

A rotina do salto compõe-se de quatro partes; cada uma delas é executada a seu tempo, quando a rotina de movimentação é chamada.

No último artigo, mostramos que o processador é direcionado para a rotina **jmp** se o conteúdo da posição de memória 57335 não é zero. Explicamos também como examinar o teclado para detectar qualquer pressão sobre as teclas M e N.



- AS QUATRO PARTES DA ROTINA DE SALTO
- SALTO VERTICAL E SALTO DIAGONAL
- EFEITOS SONOROS
- AJUSTAMENTO DOS APONTADORES DE TELA
- WILLIE ESTÁ NO AR
- DE VOLTA AO SOLO
- FIM DO SALTO

Se apenas a tecla N foi pressionada e Willie deve saltar, a posição de memória 57335 é carregada com 1, que irá direcionar o processador para a rotina aqui apresentada na próxima vez que o programa principal — que controla o jogo — chamar a rotina de movimentação. Caso as teclas M e N tenham sido pressionadas, Willie deve saltar para a frente (em diagonal); 129 é, então, carregado em 57335. Esta situação será devidamente analisada no próximo artigo da série *Avalanche*.

Por hora, imagine apenas que a rotina de movimentação de Willie constatou que houve uma pressão sobre a tecla N, na última vez que foi chamada. Agora ela voltou a ser chamada, o que fez o processador direcionar-se exatamente para nossa rotina. Depois da emissão do som que caracteriza o início de sua execução...

O conteúdo da posição de memória 57335 é carregado no acumulador outra vez. Embora ele já estivesse ali quando esta rotina foi chamada, é provável que a rotina **BEEP** tenha utilizado o acumulador para realizar o efeito sonoro do salto. Assim, convém carregar o valor em A novamente.

Depois, o conteúdo de A é comparado com 1. Naturalmente, esse valor está presente; em conseqüência, a próxima instrução, **jr nz,mjb**, não é executada e o processador continua com **inc a**. Isto incrementa o conteúdo do acumulador para 2, valor que é colocado de volta em 57335 por **ld (57335),a**. Incrementa-se, assim, o contador de salto, para que a rotina de movimentação acione a rotina **jmp** na próxima vez que for chamada. Nesse ponto, porém, a primeira parte do salto será deixada de lado; o processador executa a segunda parte. Como você deve ter reparado, cabe à instrução **cp 1**, seguida da instrução **jr nz,mjb**, fazer o processador saltar para a segunda parte.

A posição de Willie na tela é então carregada dos endereços 57332 e 57333 para o par de registros HL. O valor 32 é subtraído dessa posição e colocado em DE. Com a execução de **sbc hl, de**, o apontador se move para a posição imediatamente acima de Willie na tela. O resul-

tado é carregado de volta nos endereços 57332 e 57333, que contêm a posição do personagem na tela.

Embora essa posição tenha sido colocada nos endereços, ela ainda permanece no par HL. Todos os **ld** são essencialmente operações de cópia que colocam o valor na nova posição ou registro, mantendo esse valor na posição ou registro de origem. No nosso caso, isso é importante, pois, para chamar a rotina de impressão de bloco em 58970, a posição de impressão na tela precisa estar no par de registros HL.

O par BC, com o apontador de dados para a primeira figura de Willie saltando, é carregado em 57048. Esses dados foram apresentados em um artigo anterior de *Avalanche* e podem ser lidos numa tabela de dados na memória.

A é carregado com 40 — código de azul sobre ciano, a cor de Willie — e DE, com 259. Isso faz com que a rotina de impressão de bloco imprima um bloco um por três — 259 carrega 1 no registro D e 3 no registro E; 1 x 256 + 3 = 259. A rotina de impressão de bloco em 58970 é então chamada e imprime a figura de Willie saltando.

Em seguida, o processador retorna da rotina.

WILLIE ESTÁ NO AR

A próxima pàrte da rotina de salto é chamada quando se aciona novamente a rotina de movimentação. Mais uma vez, essa seção começa verificando se a posição de memória 57335 contém 2 —

o que acontecerá se ela estiver sendo chamada pela segunda vez.

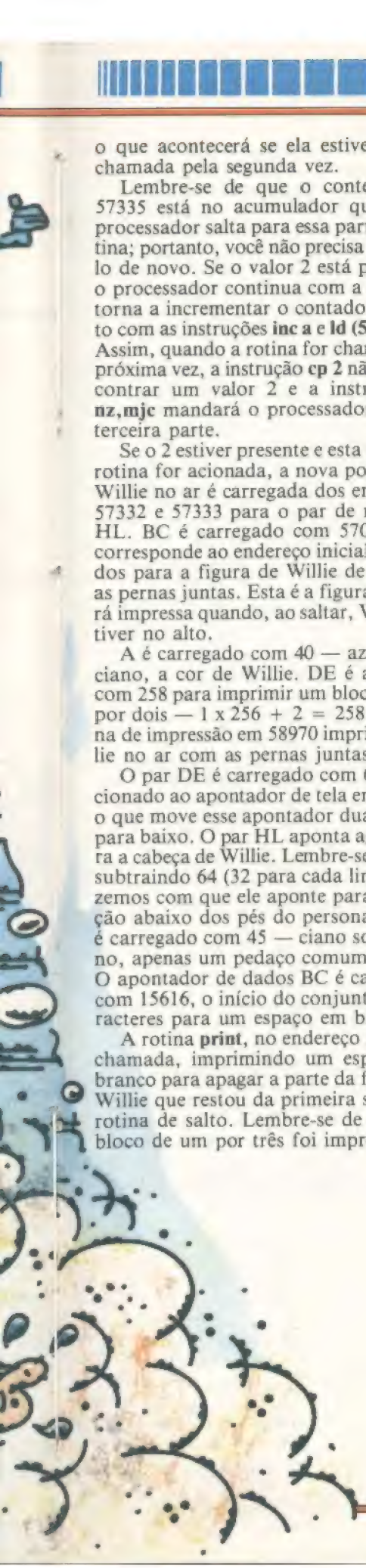
Lembre-se de que o conteúdo de 57335 está no acumulador quando o processador salta para essa parte da rotina; portanto, você não precisa carregá-lo de novo. Se o valor 2 está presente, o processador continua com a rotina e torna a incrementar o contador de salto com as instruções **inc a** e **ld (57335),a**. Assim, quando a rotina for chamada na próxima vez, a instrução **cp 2** não irá encontrar um valor 2 e a instrução **jr nz,mjc** mandará o processador para a terceira parte.

Se o 2 estiver presente e esta parte da rotina for acionada, a nova posição de Willie no ar é carregada dos endereços 57332 e 57333 para o par de registros HL. BC é carregado com 57000, que corresponde ao endereço inicial dos dados para a figura de Willie de pé com as pernas juntas. Esta é a figura que será impressa quando, ao saltar, Willie estiver no alto.

A é carregado com 40 — azul sobre ciano, a cor de Willie. DE é ajustado com 258 para imprimir um bloco de um por dois — 1 x 256 + 2 = 258. A rotina de impressão em 58970 imprime Willie no ar com as pernas juntas.

O par DE é carregado com 64 e adicionado ao apontador de tela em HL — o que move esse apontador duas linhas para baixo. O par HL aponta agora para a cabeça de Willie. Lembre-se de que, subtraindo 64 (32 para cada linha), fazemos com que ele aponte para a posição abaixo dos pés do personagem. A é carregado com 45 — ciano sobre ciano, apenas um pedaço comum de céu. O apontador de dados BC é carregado com 15616, o início do conjunto de caracteres para um espaço em branco.

A rotina **print**, no endereço 58217, é chamada, imprimindo um espaço em branco para apagar a parte da figura de Willie que restou da primeira seção da rotina de salto. Lembre-se de que um bloco de um por três foi impresso ali, e, agora, apenas um bloco de um por dois. O processador retorna de novo, até a rotina voltar a ser chamada.



DESCENDO

Na próxima vez, o processador acionará a terceira parte da rotina de movimentação. Antes de mais nada, ele verifica se é necessário acionar a quarta parte da rotina de salto. Em seguida, o contador de salto, que ainda está no acumulador, é incrementado e armazenado novamente em 57335, para que o processador pule para a quarta parte da rotina na próxima vez.

O apontador da posição de Willie na tela é carregado de volta em HL. O apontador de dados em BC é ajustado com o mesmo valor utilizado na primeira parte da rotina de salto. Willie descerá ao chão pelo mesmo processo que subiu.

A é carregado de novo com 40 — azul sobre ciano; DE volta a ser ajustado com 259 — um por três. A rotina de impressão de bloco em 58970 é chamada e imprime Willie descendo.

O par DE é carregado com 32, valor adicionado ao par HL. O resultado é carregado de volta em 57332 e 57333. Isso move o apontador de tela uma posição para baixo.

O processador retorna agora com as posições dos dados ajustados para a próxima parte.

FIM DO SALTO

Mais uma vez a rotina começa com um **cp**, para verificar se esta é a parte necessária da rotina de salto. Mas, agora, estamos diante da última seção da rotina que faz Willie pular. Para que o processador chegue até aqui, o conteúdo do acumulador deve ser 4 (parte que finaliza o salto) ou 129, valor que indica que Willie irá saltar para a frente. Nesse caso, **jr nz,mfj** manda o processador para outra instrução de retorno, no final deste programa. O retorno será apagado mais tarde, servindo para indicar o local onde se inicia a rotina de salto à frente, que será fornecida no próximo artigo de *Avalanche*. Desta vez, entretanto, nenhum teste será feito e não precisaremos incrementar o contador de salto. É necessário que se ajuste o contador com 0, para que, quando a rotina de movimentação for novamente chamada, o processador execute a parte dada no artigo anterior e verifique se alguma tecla foi pressionada. Isso é feito por **ld a,0** e **ld (57335),a**.

A posição de Willie volta a ser carregada de 57332 e 57333 para o par de registros HL, onde 32 é subtraído através do par DE, fazendo o apontador se mover uma posição acima na tela.

BC é carregado com o endereço de um espaço vazio e A, com o código de ciano sobre ciano. A rotina **print**, em 58217, é chamada de novo. Apaga-se, assim, o caractere adicional ocupado por Willie, impresso na terceira parte da rotina de salto.

A instrução **jp 59153** faz o processador voltar para o início da rotina de movimentação, dada no artigo anterior. Como o contador de salto contém agora o valor 0, o processador executa a rotina que imprime Willie parado, apagando o que restou da antiga figura.

A última instrução **ret** da listagem tem a função de prevenir que ocorra um erro no programa, caso a rotina de salto para a frente venha a ser chamada. Essa instrução será apagada na próxima parte do programa.

```
10   ORG 20140
20   JUM JSR CLICK
30   LDA 18261
40   CMPA #1
50   BNE MJA
60   INC 18261
70   LDX 18249
80   PSHS X
90   LEAX 256,X
100   LDU #1536
110   JSR CHARPR
120   PULS X
130   LEAX -256,X
140   STX 18249
150   LDU #17814
160   JSR CHARPR
170   LEAX 254,X
180   LDU #17846
190   JSR CHARPR
200   RTS
210   MJA CMPA #2
220   BNE MJB
230   INC 18261
240   LDX 18249
250   LEAX -256,X
260   LDU #17870
270   JSR CHARPR
280   LEAX 254,X
290   JSR CHARPR
300   LEAX 254,X
310   JSR CHARPR
320   RTS
330   MJB CMPA #3
340   BNE MJC
350   INC 18261
360   LDX 18249
370   LEAX -256,X
380   LDU #1536
390   JSR CHARPR
400   LEAX 254,X
410   LDU #17926
```

```
420  JSR CHARPR
430  LEAX 254,X
440  LDU #17958
450  JSR CHARPR
460  LEAX 254,X
470  LDU #17990
480  JSR CHARPR
490  RTS
500  MJC CMPA #4
510  BNE MFJ
520  CLR 18261
530  LDX 18249
540  PSHS X
550  LDU #1536
560  JSR CHARPR
570  PULS X
580  LEAX 256,X
590  STX 18249
600  LDU #17774
610  JSR CHARPR
620  LEAX 254,X
630  JSR CHARPR
640  RTS
650  MFJ RTS
660  CHARPR EQU 19402
670  CLICK EQU 20847
```

A primeira coisa que a rotina **JUM** (ou de salto) faz é chamar a sub-rotina **CLICK**, que toca uma das notas do efeito sonoro do salto. Se você ainda não tiver digitado essa rotina, adicione um retorno em 20847 por meio da instrução **POKE**, com 57 nesse endereço. Assim, não haverá erro quando o programa for testado.

PULANDO POR ETAPAS... JÁ!

O acumulador é carregado com o conteúdo da posição de memória 18261, que contém a variável do salto. Inicialmente, ela está carregada com 1 ou 129 — dependendo de qual tecla foi pressionada — indicando, portanto, se Willie deve pular verticalmente ou dar um salto à frente. Cada um desses tipos de pulo é dividido em quatro etapas, executadas uma por vez, quando a rotina de movimentação é chamada. Para garantir o acionamento da etapa correta, a variável do salto é incrementada na execução de cada parte.

Se a variável do salto é 0, utiliza-se a seção da rotina que examina o teclado para ver se uma ordem de pular foi dada. Caso a variável de movimentação contenha qualquer outro valor que não seja 0, o processador vem para esta parte do programa e executa uma das rotinas de salto.

Se for 1, a primeira parte da rotina é acionada. Durante sua execução, a variável de salto é incrementada, para que, na próxima vez que a rotina de movimentação for chamada, essa seção seja evitada e o processador passe para a parte dois. Assim, sempre que a rotina de movimentação for acionada, ela chamará a próxima parte — até que a última (a quarta) seja executada e a variável do salto volte a ser ajustada com 0.

Se a última rotina ajustou a variável do salto com 129, esta parte é ignorada. A rotina do salto para a frente, que daremos no próximo artigo de *Avalanche*, é então acessada.

UM!

O conteúdo do acumulador é comparado com 1. Se 1 não está presente, o processador passa para **MJA**. Caso contrário, o processador continua com esta parte da rotina.

A primeira coisa que ela faz é incrementar a variável em 18261, deixando-a pronta para a próxima vez.

X é carregado com a posição de Willie, que está no endereço 18249 e é armazenado temporariamente na pilha. O apontador em X é incrementado com 256, movendo-se um caractere para baixo. U é carregado com 1536, o endereço do canto superior esquerdo da tela. O processador chama então a rotina **CHARPR**, que imprime um bloco de céu azul sobre a metade inferior de Willie. Se não fizermos isso, as pernas de Willie aparecerão em sua posição anterior.

A posição de Willie é novamente guardada na pilha e movida um caractere para cima pela subtração de 256. O resultado é armazenado de volta em 18249. U é carregado com 17814, endereço inicial para os dados de Willie com as pernas abertas. Em seguida, a rotina **CHARPR** é chamada para imprimir a metade superior.

O apontador de tela é incrementado com 254, passando a apontar para o início da metade inferior de Willie. U é recarregado e a rotina **CHARPR** volta a ser chamada para imprimir sua metade inferior.

O processador retorna depois de ter imprimido Willie com as pernas abertas um caractere acima do chão.

DOIS!

Quando a rotina de movimentação de Willie for novamente chamada, o número no endereço 18261 será 2. Portanto, o processador irá pular a primeira parte da rotina publicada no artigo anterior e a parte desta rotina que foi dada até agora. Mas, quando ele saltar de novo para **MJA**, o conteúdo de 18261, que ainda está no acumulador, será comparado com 2. Na chamada seguinte da rotina de movimentação, quando o conteúdo do endereço 18261 tiver sido incrementado mais uma vez, a instrução **BNE MJB** mandará o processador verificar se seu valor é 3, 4, 129, 130, 131 ou 132. Desta vez ele executará a próxima seção da rotina, imprimindo a figura de Willie no segundo estágio do pulo.

Essa seção também começa incrementando o conteúdo de 18261, acertando-o para a próxima chamada da rotina de salto. Depois, a nova posição de Willie é carregada de 18249 para X e decrementada de 256 — o que faz o personagem se mover um caractere acima na tela. Mas agora o resultado não é colocado de volta em 18261. Willie está no alto de seu pulo.

U é carregado com 17870, endereço inicial dos dados da nova figura de Willie. Desta vez, ele tem três caracteres de altura, impressos em três posições — **CHARPR** precisa ser chamada três vezes e X é decrementado com 254 para mover o apontador uma linha abaixo em cada chamada.

Feito isso, o processador retorna.

TRÊS!

Quando a rotina de movimentação volta a ser chamada, o conteúdo de 18261 é 3 e o processador executa a rotina **MJB**. Ela começa, mais uma vez, incrementando 18261, carregando 18249 em X e subtraindo 256.

Só que agora a figura anterior de Willie deve ser apagada. U é carregado com 1536, o endereço do canto superior esquerdo da tela, e a rotina **CHARPR** é chamada, imprimindo a cabeça de Willie na tela. O apontador X é incrementado para indicar a posição inferior na tela e U é carregado com o endereço dos dados para a impressão do resto da figura.

QUATRO!

Se nosso personagem está pulando na vertical, mais cedo ou mais tarde o processador irá executar **MJC**. As instruções **CMPA #4** e **BNE** chamarão **MJF** apenas quando Willie for saltar na diagonal. Mas aquela pequena rotina será sempre executada.

Como chegamos à quarta e última parte da rotina que faz Willie pular verticalmente, o conteúdo da posição 18261 não é incrementado. A instrução **CLR 18261** ajusta essa variável com 0; a ro-

tina de movimentação de Willie é, portanto, executada e verifica-se, pelo exame do teclado, se ele deve pular de novo.

A posição de Willie é carregada em X e colocada na pilha de máquina para armazenamento temporário. Como a figura tem apenas dois caracteres de altura quando está parada ou andando, e três, quando está pulando, o caractere mais alto tem que ser apagado de novo. Note que, agora, Willie está numa posição intermediária, usada para tornar o pulo mais suave.

U é carregado com 1536, que aponta para uma parte do céu no topo da tela. Quando é chamada, a rotina **CHARPR** apaga a cabeça da figura anterior.

A posição de Willie é recuperada da pilha e adicionada a 256, o que faz a figura se mover um caractere abaixo na tela. Lembre-se de que sua posição tinha sido decrementada com 256 no início desta rotina, quando Willie começou o salto. Assim, quando o novo valor é armazenado em 18261, o personagem simplesmente volta para a posição anterior ao do pulo.

U é então carregado com 17774, endereço inicial dos dados para a figura de Willie com as pernas juntas. **CHARPR** é chamada duas vezes e X é adicionado com 254, imprimindo a figura completa de Willie na tela.

```
10    org 54854
20    pl ld a,(-5202)
30    cp 1
40    jr nz,pb
50    inc a
60    ld (-5202),a
70    ld hl,(-5205)
80    ld de,32
90    sbc hl,de
100    ld (-5205),hl
110    ld de,(62407)
120    add hl,de
130    ld a,10
140    push hl
150    call 77
160    pop hl
170    ld de,32
180    add hl,de
190    ld a,11
200    push hl
210    call 77
220    pop hl
230    ld de,32
240    add hl,de
250    ld a,255
260    call 77
270    ret
280    pb cp 2
290    jr nz,pc
300    inc a
310    ld (-5202),a
320    ld hl,(-5205)
330    ld de,(62407)
340    add hl,de
350    ld a,0
360    push hl
370    call 77
380    pop hl
390    ld de,32
400    add hl,de
410    ld a,1
420    call 77
430    ret
440    pc cp 3
450    jr nz,pd
460    inc a
470    ld (-5202),a
480    ld hl,(-5205)
490    ld de,(62407)
500    add hl,de
510    ld a,10
520    push hl
530    call 77
540    pop hl
550    ld de,32
560    add hl,de
570    ld a,11
580    call 77
590    ret
600    pd cp 4
610    jr nz,mp
620    ld a,0
630    ld (-5202),a
640    ld hl,(-5205)
650    push hl
660    ld de,(62407)
670    add hl,de
680    ld a,255
690    call 77
700    pop hl
710    ld de,32
720    add hl,de
730    ld (-5205),hl
740    jp 54603
750    mp ret
760    end
```

SUBINDO

A rotina de salto é dividida em quatro partes; cada uma delas é executada separadamente quando a rotina de movimentação de Willie é chamada. No último artigo, vimos como o processador é direcionado para a rotina **pl** quando o conteúdo da posição de memória – 5202 não é zero. Vimos também como examinar o teclado para descobrir se as teclas de controle M e N foram pressionadas, individualmente ou ao mesmo tempo.

Se a tecla M foi pressionada e Willie deve saltar, a posição de memória – 5202 é carregada com 1. Isso fará com que o processador seja direcionado para esta rotina na próxima vez que a rotina principal do jogo chamar a rotina de movimentação.

Se M e N foram pressionadas, Willie deve saltar na diagonal e 129 é carregado em – 5202. Este caso será examinado no próximo artigo.

Por enquanto, suponhamos que a rotina de movimentação detectou que a tecla N foi pressionada na última vez que foi chamada. Agora, ela voltou a ser chamada, o que fez o processador direcionar-se para a rotina do salto, cuja execução se inicia...

O conteúdo da posição – 5202 da memória é carregado novamente no acumulador. Em seguida, o valor em A é comparado com 1. Como a suposição feita exige que o valor seja 1, a instrução seguinte, **jr nz,pb**, não é executada e o processador continua com **inc a**. Com isso, o conteúdo do acumulador passa a ser 2 — valor que é colocado de volta em – 5202 pela instrução **ld (** – 5202), a. Incrementa-se, assim, o contador de salto para que, na próxima vez que a rotina de movimentação for chamada, ela acione a rotina **pl**. Neste ponto, porém, a primeira parte do salto é deixada de lado e o processador executa a segunda.

A posição de Willie na tela é carregada dos endereços – 5205 e – 5204 no par HL. O número 32 é subtraído dessa posição e colocado em DE; a instrução **sbc hl, de** é executada. Isso coloca o apontador de posição um caractere acima de Willie na tela. O resultado é carregado de volta nos endereços que contêm a posição de Willie, – 5205 e – 5204.

Embora esse valor tenha sido colocado nos endereços, ele permanece no par HL. Todos os **ld** são essencialmente operações de cópia, que colocam o valor na nova posição ou registro, mantendo esse valor na posição ou registro de origem. No nosso caso isso é importante, pois o valor em HL é somado ao endereço inicial da Tabela de Nomes da VRAM, passando a apontar para a posição de Willie na TN.

O acumulador é carregado com 10,

código do primeiro padrão da figura de Willie subindo, com as pernas abertas. O apontador no par HL é guardado na pilha e a rotina 77 da ROM é chamada. Essa rotina escreve o valor de A no endereço da VRAM apontado por HL — o que equivale, aqui, a colocar o padrão na tela.

O apontador é recuperado da pilha e somado com 32, voltando-se para uma posição abaixo da anterior. A é carregado com 11, código do segundo padrão da figura de Willie. O apontador é novamente guardado na pilha e a rotina 77 da ROM é chamada.

A figura de Willie saltando já está completa. Falta só apagar uma parte da figura anterior. Para isso, recupera-se o apontador da pilha e adiciona-se 32, fazendo-o apontar para o padrão que se pretende eliminar. O código do padrão de céu, 255, é colocado em A e a rotina 77 é chamada, apagando parte da figura anterior. Em seguida, o processador retorna.

WILLIE ESTÁ NO AR

A segunda parte da rotina de salto é chamada assim que se aciona a rotina de movimentação. Como as outras partes, ela começa verificando se a posição de memória −5202 contém 2 — o que acontece se esta parte ainda não foi executada.

Lembre-se de que o conteúdo de −5202 está no acumulador quando o processador aciona esta parte — portanto, não é necessário carregá-lo outra vez. Se o valor 2 está presente, o processador continua com a rotina e incrementa o contador de salto com as instruções **inc a** e **ld (−5202), a**. Quando esta parte for acessada na próxima vez, **cp 2** não irá encontrar o valor 2 e **jr nz,pc** mandará então o processador para a terceira parte do salto.

Se o valor 2 estiver presente e esta parte da rotina foi acionada, a atual posição de Willie é transferida dos endereços −5205 e −5204 para o par de registros HL. O par DE é carregado com o endereço inicial da TN da VRAM e somado em HL. O acumulador é carregado com 0, o código do primeiro padrão da figura de Willie parado. O apontador em HL é guardado na pilha e a rotina 77 da ROM é chamada.

O apontador é recuperado da pilha e somado com 32 por meio de DE, o que o faz se mover uma posição para baixo. A é carregado com 1, o código do segundo padrão da figura. A rotina 77 é chamada de novo. Neste ponto, Willie está impresso no ar com as pernas juntas. Observe que agora não é preciso apagar nada, pois a figura foi impressa sobre a anterior.

O processador retorna da rotina até a próxima chamada.

DESCENDO

Desta vez, o processador executa a terceira parte da rotina de salto. Primeiro, ele verifica se é necessário acionar a quarta parte do salto.

Em seguida, o contador de salto, que ainda está no acumulador, é incrementado e devolvido a −5202, para que o processador passe para a quarta parte na próxima vez.

O apontador de Willie na tela é carregado no par HL. O par DE é carregado com o endereço inicial da TN da VRAM e somado em HL. Em seguida, a mesma figura da primeira parte é impressa de maneira idêntica, utilizando a rotina 77 da memória ROM. O processador retorna desta parte.

FIM DO SALTO

Mais uma vez a rotina inicia com um **cp**, verificando se esta é a parte necessária da rotina de salto. Para que o processador chegue até aqui, o conteúdo do acumulador deve ser 4 ou 129. Se for 4, esta é a última parte da rotina do salto; se for 129, Willie irá saltar na diagonal. Nesse caso, a instrução **jr nz,mp** manda o processador para a instrução de retorno do final deste programa. Essa instrução será apagada pela rotina de salto diagonal, que apresentaremos no próximo artigo de *Avalanche*.

Nenhum teste será feito, nem precisaremos incrementar o contador de salto. É necessário, porém, ajustar esse contador com zero, para que, quando a rotina de movimentação for chamada de novo, o processador execute a primeira parte, dada no artigo anterior, que verifica se alguma tecla foi pressionada. Isso é feito pelas instruções **ld a,0** e **ld (−5202),a**.

A posição de Willie é mais uma vez carregada no par HL, sendo armazenada na pilha. O endereço inicial da TN da VRAM é colocado em DE e somado ao valor de HL. O acumulador é carregado com 255, o código do padrão de céu. A rotina 77 da ROM é chamada, apagando parte da figura anterior.

A posição de Willie é recuperada da pilha para HL, somada com 32 através de DE e, finalmente, devolvida aos endereços −5205 e −5204.

A primeira rotina de movimentação de Willie é chamada por **jp 54603**. Não se preocupe com o que restou de sua figura, pois nossa rotina apagará todos os vestígios indesejáveis.

O último **ret** da listagem é colocado apenas para prevenir possíveis erros, que ocorreriam se a rotina de salto diagonal fosse chamada sem estar presente na memória. Essa instrução será apagada pela parte do programa dada no próximo artigo de *Avalanche*.

MODELOS DA REALIDADE

Para ganhar a quina da loto, basta escolher cinco números ao acaso — os cinco números certos, é claro. Veja aqui algumas dicas estatísticas para programar eventos aleatórios.

Os engenheiros constroem modelos em miniatura, para testar suas propriedades. Quando fazem isso, estão simulando um sistema. O programa do artigo da página 1121 executa algo bem parecido: ele simula um evento, oferecendo resultados compatíveis com a realidade. Veremos agora como combinar os princípios da simulação e da estatística para resolver problemas de ordem prática e para acrescentar a nossos jogos uma dose de realidade.

TIPOS DE MODELO

Um modelo é uma representação simplificada de um sistema real, por meio da qual se procura explicar ou prever as características e o comportamento desse sistema. Existem três tipos de modelo — icônico, análogo e simbólico —, nem todos úteis aos usuários de microcomputadores.

Modelos icônicos não possuem partes móveis. Maquetes de estradas e prédios incluem-se nessa categoria. Ao projetar a construção de uma ponte sobre um estuário, por exemplo, engenheiros civis providenciam uma maquete da ponte e da área adjacente, a fim de estudar os efeitos da maré, da correnteza e do vento. Embora se utilizem computadores para interpretar os resultados, a necessidade de coletar dados através de um experimento torna a simulação dispensável.

Modelos análogos representam um fenômeno — ou quantidade — por outro. É o caso de um economista que, para representar o fluxo de dinheiro na economia mundial, utiliza o fluxo de água de um conjunto de tanques cheios até diferentes níveis. Esse tipo de representação pode ser feito eletronicamente — empregando-se computadores analógicos e não digitais.

O terceiro tipo de modelo, o simbólico, é o que apresenta particular interesse para o usuário do micro.

INCERTEZAS

Num modelo simbólico, representa-se determinado sistema por meio de uma fórmula matemática. Para estudar a distância **S** percorrida por um carro que se desloca à velocidade constante **V** por um certo período de tempo **T**, um modelo adequado seria **S = V*T**. Essa equação é chamada *determinística*, pois não comporta nenhum elemento de incerteza ou acaso.

Existem, contudo, eventos que não podem ser previstos com tal grau de precisão. Temos um bom exemplo disso no programa apresentado no artigo da página 776, que simula o lançamento de uma moeda. Modelos que descrevem fenômenos que incluem um elemento de acaso são chamados *estocásticos*. Este é o tipo de evento que se costuma simular, especialmente em jogos.

Algumas variáveis só podem assumir valores discretos como 0, 1, 2, 3... O número de gols em uma partida de futebol pertence a essa categoria. Quando simulamos variáveis aleatórias como essa, precisamos sempre levar em conta que certos processos aparentemente diferentes têm, na realidade, a mesma estrutura.

Tomemos como exemplo o programa que simula o lançamento da moeda. Ele poderia ser facilmente modificado a fim de fornecer os resultados de um jogo de dados ou da cobrança de pênaltis. Em todas essas circunstâncias há um certo número de eventos independentes, cada qual com uma probabilidade definida. No caso de lançarmos uma moeda quatro vezes, vários resultados são possíveis: quatro caras, três caras e uma coroa, duas caras e duas coroas, uma cara e três coroas, quatro coroas. Cada um dos resultados, porém, tem uma probabilidade definida. Os processos desse tipo, conhecidos como *processos de Bernoulli*, são formados por variáveis alea-

- TIPOS DE MODELO
- INCERTEZAS
- PROBABILIDADE E COMPORTAMENTO ALEATÓRIO
- VARIÁVEIS ALEATÓRIAS
- SIMULAÇÕES
- DISTRIBUIÇÃO UNIFORME
- DISTRIBUIÇÃO NORMAL
- DISTRIBUIÇÃO EXPONENCIAL
- COMPARAÇÃO DE EVENTOS

tórias discretas. Eles não se prestam, por exemplo, para descrever o número de acidentes do campeonato mundial de Fórmula-1 ou o número de chamadas telefônicas que chegam a uma central em meia hora. Nessas situações, os eventos ocorrem ao acaso durante um certo período de tempo e não são o resultado de experimentos sucessivos. Tais processos são denominados *processos de Poisson*.

Para obter uma distribuição de Poisson, digite este programa:

```
10 BORDER 0: PAPER 0: INK 7:
CLS
20 POKE 23658,0
40 PRINT AT 1,6; INVERSE 1;"S
IMULACAO DE POISSON ": PRINT
: PRINT
50 INPUT "QUAL E O VALOR MEDI
O ";a
70 INPUT "TAMANHO DA AMOSTRA
";n
80 FOR i=1 TO n
90 LET c=0: LET t=1
100 LET s=EXP (-a)
110 LET t=t*(RND*1)
120 IF t<=s THEN  PRINT "
 ";c;: GOTO 150
130 LET c=c+1
140 GOTO 110
150 NEXT i
160 INPUT " OUTRA AMOSTRA  (S/
N) ?";q$
170 IF q$="s" THEN  GOTO 10
180 STOP
```

```
10 R=RND(-TIME)
30 CLS:WIDTH 40
40 PRINT "Distribuão de Poisson
":PRINT:PRINT
50 INPUT "Qual é o valor da méd
ia";A
60 PRINT
70 INPUT "Tamanho da amostra";N
80 FOR I=1 TO N
90 C=0:T=1
100 S=EXP(-A)
110 T=T*RND(1)
120 IF T<=S THEN PRINT LEFT$(ST
R$(C)+"        ",8);:GOTO 150
130 C=C+1
140 GOTO 110
150 NEXT I
160 PRINT:INPUT "Outra amostra
(Y/N)";G$
170 IF G$="Y" THEN 30
180 END
```

```
30  HOME
40  PRINT  TAB( 9);"DISTRIBUICA
O DE POISSON": PRINT : PRINT
50  INPUT "QUAL O VALOR DA MEDI
A ";A
60  PRINT
70  INPUT "TAMANHO DA AMOSTRA "
;N
80  FOR I = 1 TO N
90 C = 0:T = 1
100 S =  EXP ( - A)
110 T = T *  RND (1)
120  IF T <  = S THEN  PRINT  L
EFT$ ( STR$ (C) + "        ",8);
: GOTO 150
130 C = C + 1
140  GOTO 110
150  NEXT I
160  PRINT : INPUT "OUTRA AMOST
RA (S/N) ";G$
170  IF G$ = "S" THEN 30
180  END
```

```
30 CLS
40 PRINT @6,"simulacao de poiss
on":PRINT:PRINT
50 INPUT"QUAL E O VALOR MEDIO "
;A
60 PRINT
70 INPUT"TAMANHO DA AMOSTRA ";N
80 FOR I=1 TO N
90 C=0:T=1
100 S=EXP(-A)
110 T=T*RND(0)
120 IF T<=S THEN PRINT LEFT$(ST
R$(C)+"        ",8);:GOTO 150
130 C=C+1
140 GOTO 110
150 NEXT I
160 PRINT:INPUT"OUTRA AMOSTRA (
S/N) ";G$
170 IF G$="S" THEN 30
180 END
```

Ao ser executado, o programa solicita o valor médio, bem como o número de elementos da amostra. A parte principal do programa (linhas 80 a 150) usa uma equação para gerar as variáveis de Poisson. A variável **S** recebe o valor de **e** (uma constante matemática) elevada ao valor de **A** (valor da média). A linha 110 escolhe um número aleatório entre **0** e **T**. A linha 120 compara as variáveis **S** e **T** para decidir se imprime ou não uma variável **C**.

Suponhamos que você queira criar um jogo onde uma nave espacial enfrenta uma chuva de meteoritos. Se o número médio de "colisões" em um período de um minuto é dois, e você precisa de cinco simulações de um minuto, o computador poderá fornecer o resultado 2, 3, 2, 1, 0 como amostra do número de choques que a nave sofrerá em um minuto. Cada valor não é muito diferente de 2 (a média) e há cinco valores — o tamanho da amostra.

Poderíamos usar a mesma técnica para simular os resultados de uma série de partidas de futebol? É claro que sim. Estaremos lidando com uma variável discreta, pois o número de gols em uma partida só pode assumir valores inteiros. Gols são eventos que ocorrem ao acaso, em determinado espaço de tempo — e não o resultado de um certo número

de experimentos repetidos. Portanto, nesse caso, um processo de Poisson é adequado.

Além do número de partidas, precisaremos escolher a média de gols por partida. Este é o momento de buscar dados no mundo real. A tabela abaixo mostra a média de gols por partida nos campeonatos da primeira divisão inglesa entre 1977 e 1983.

Primeira divisão
Média de gols por partida

Temporada	Time da casa	Time visitante
77-78	1.6039	1.0606
78-79	1.5498	1.0844
79-80	1.6925	0.9481
80-81	1.6364	1.0216
81-82	1.4545	1.0844
82-83	1.7532	0.9822

Baseados nesses valores, podemos estimular uma média de 1.7 gols por partida para o time da casa e de 1.0 para o time visitante. Dividindo esses números por dois, obteremos as médias de 0.85 e 0.5 para meio tempo de jogo. De posse desses dados, fica fácil elaborar o programa:

```
20 DIM a$(25,10): DIM h$(25,
10): DIM h(50): DIM a(50):
DIM f(50): DIM g(50)
30 BORDER 0: PAPER 0: INK 7:
CLS
50 PRINT AT 0,8; INVERSE 1;"
 PLACAR FINAL  "
70 INPUT " QUANTAS PARTIDAS (
1-25)? ";n
75 IF n<1 OR n>25 THEN  GOTO
 70
80 FOR i=1 TO n
90 PRINT "PARTIDA";i
100 INPUT "TIME DA CASA";h$(i)
110 INPUT "TIME VISITANTE";a$(
i)
120 NEXT i
130 PAUSE 400
140 PRINT
150 FOR i=1 TO 2*n
160 LET c=0: LET t=1
170 LET s=EXP (-.85)
180 LET t=t*(RND*1)
190 IF t<=s THEN  LET h(i)=c:
GOTO 220
200 LET c=c+1
210 GOTO 180
220 NEXT i
230 FOR i=1 TO 2*n
240 LET c=0: LET t=1
250 LET s=EXP (-.5)
260 LET t=t*(RND*1)
270 IF t<=s THEN  LET a(i)=c:
GOTO 300
280 LET c=c+1
290 GOTO 260
300 NEXT i
310 CLS
320 PRINT TAB (7);"PLACAR PRIM
EIRO TEMPO";''
330 FOR i=1 TO n
340 PRINT h$(i);h(i);TAB 20;a$
(i);a(i): PRINT
350 NEXT i
360 PRINT " QUALQUER TECLA PAR
A CONTINUAR"
370 IF INKEY$="" THEN  GOTO
 370
375 CLS
380 PRINT TAB 12;"PLACAR FINAL
"''
390 FOR i=1 TO n
400 LET f(i)=h(i)+h(n+i)
410 LET g(i)=a(i)+a(i+n)
420 PRINT h$(i);f(i);TAB 20;a$
(i);g(i): PRINT
430 NEXT i
440 INPUT " NOVAMENTE ?   (s/n
) ";g$
450 IF g$="n" THEN  GOTO 490
460 INPUT "MESMOS TIMES (s/n)
?";p$
470 IF p$="n" THEN  GOTO 70
480 GOTO 150
490 STOP
```

```
5 R=RND(-TIME)
20 DIM A$(25),H$(25),H(50),A(50
),FA(25),FH(25)
30 CLS:WIDTH 40
50 PRINT TAB(13);"SCORE FINAL"
60 PRINT:PRINT
70 INPUT"Quantas partidas (1-25
)";N
75 IF N<1 OR N>25 THEN 70
80 FOR I=1 TO N
90 PRINT "PALITO";I
100 INPUT "Time da casa";H$(I)
110 INPUT "Time de fora";A$(I)
120 NEXT I
130 FOR V=1 TO 2000:NEXT V
140 PRINT
150 FOR I=1 TO 2*N
160 C=0:T=1
170 S=EXP(-.85)
180 T=T*RND(0)
190 IF T<=S THEN H(I)=C:GOTO 22
0
200 C=C+1
210 GOTO 180
220 NEXT I
230 FOR I=1 TO 2*N
240 C=0:T=1
250 S=EXP(-.5)
260 T=T*RND(0)
270 IF T<=S THEN A(I)=C:GOTO 30
0
280 C=C+1
290 GOTO 260
300 NEXT I
310 CLS
320 PRINT TAB(9);"Score temporá
rio":PRINT:PRINT
330 FOR I=1 TO N
340 PRINT H$(I);H(I),A$(I);A(I)
350 NEXT I
360 P..INT "Aperte qualquer tecl
a"
370 IF INKEY$="" THEN 370
380 CLS
390 FOR I=1 TO N
400 FH(I)=H(I)+H(N+I)
410 FA(I)=A(I)+A(N+I)
420 PRINT H$(I);FH(I),A$(I);FA(
I)
430 NEXT
440 PRINT:INPUT"OUTRA VEZ (S/N)
";G$
450 IF G$="N" THEN END
460 INPUT "Mesmos times (S/N)";
P$
470 IF P$="N" THEN 70
480 GOTO 150
```

```
20  DIM A$(25),H$(25),H(50),A(5
0),FA(25),FH(25)
30  HOME
50  PRINT  TAB( 13);"SCORE FINA
L "
60  PRINT : PRINT
70  INPUT "QUANTAS PARTIDAS (1-
25) ";N
75  IF N < 1 OR N > 25 THEN 70
80  FOR I = 1 TO N
90  PRINT "PARTIDA ";I
100  INPUT "TIME DA CASA ";H$(I
)
110  INPUT "TIME DE FORA ";A$(I
)
120  NEXT I
130  FOR V = 1 TO 2000: NEXT V
140  PRINT
150  FOR I = 1 TO 2 * N
160 C = 0:T = 1
170 S =  EXP ( - .85)
180 T = T *  RND (1)
190  IF TT <  = S THEN H(I) = C
: GOTO 220
200 C = C + 1
210  GOTO 180
220  NEXT I
230  FOR I = 1 TO 2 * N
240 C = 0:T = 1
250 S =  EXP ( - .5)
260 T = T *  RND (1)
270  IF T <  = S THEN A(I) = C:
 GOTO 300
280 C = C + 1
290  GOTO 260
300  NEXT I
310  HOME
320  PRINT "SCORE PARCIAL ": PR
INT : PRINT
330  FOR I = 1 TO N
340  PRINT H$(I);H(I),A$(I);A(I
)
350  NEXT I
360  PRINT "APERTE QUALQUER TEC
LA "
370  GET G$
375  HOME
380  PRINT  TAB( 13);"SCORE FIN
AL": PRINT : PRINT
390  FOR I = 1 TO N
400 FH(I) = H(I) + H(N + I)
410 FA(I) = A(I) + A(N + I)
420  PRINT H$(I);FH(I),A$(I);FA
(I)
```

```
430  NEXT
440  PRINT : INPUT "OUTRA VEZ (
S/N) ";P$
450  IF P$ = "N" THEN  END
460  INPUT "MESMOS TIMES (S/N)
";G$
470  IF G$ = "N" THEN 70
480  GOTO 150
```

```
20 DIM A$(25),H$(25),H(50),A(50
),FA(25),FH(25)
30 CLS
50 PRINT @11,"placar final"
60 PRINT:PRINT
70 INPUT"QUANTAS PARTIDAS (1-25
) ";N
80 FOR I=1 TO N
90 PRINT"PARTIDA ";I
100 INPUT"TIME DA CASA ";H$(I)
110 INPUT"VISITANTE ";A$(I):PRI
NT
120 NEXT I
130 FOR V=1 TO 2000:NEXT V
140 PRINT
150 FOR I=1 TO 2*N
160 C=0:T=1
170 S=EXP(-.85)
180 T=T*RND(0)
190 IF T<=S THEN H(I)=C:GOTO220
200 C=C+1
210 GOTO 180
220 NEXT I
230 FOR I=1 TO 2*N
240 C=0:T=1
250 S=EXP(-.5)
260 T=T*RND(0)
270 IF T<=S THEN A(I)=C:GOTO300
280 C=C+1
290 GOTO 260
300 NEXT I
310 CLS
320 PRINT @5,"placar primeiro t
empo":PRINT:PRINT
330 FOR I=1 TO N
340 PRINT H$(I);H(I);A$(I);A(I)
350 NEXT I
360 PRINT" QUALQUER TECLA PARA
CONTINUAR"
370 IF INKEY$="" THEN 370
375 CLS
380 PRINT @10,"placar final":PR
INT:PRINT
390 FOR I=1 TO N
400 FH(I)=H(I)+H(N+I)
410 FA(I)=A(I)+A(N+I)
420 PRINT H$(I);FH(I),A$(I);FA(
I)
430 NEXT
440 PRINT:INPUT"NOVAMENTE  (S/N
) ";G$
450 IF G$="N" THEN END
460 INPUT"MESMOS TIMES  (S/N) "
;P$
470 IF P$="N" THEN 70
480 GOTO 150
```

Da linha 20 a 120 o programa solicita as informações iniciais. Para evitar muita digitação, podemos substituir os nomes dos times por letras. As linhas 150 a 300 usam o algoritmo de Poisson, idêntico ao do programa anterior, para gerar os resultados parciais e finais dos jogos. As linhas 320 a 350 cuidam da saída dos resultados parciais e as linhas 390 a 430, dos resultados finais.

A simulação de Poisson funciona bem com variáveis discretas. Algumas vezes, porém, precisamos utilizar variáveis fracionárias ou contínuas. Em uma simulação de competição de salto em distância, por exemplo, podemos obter uma medida qualquer entre sete e nove metros — o comprimento do salto é uma variável aleatória, contínua.

Embora a função **RND** seja adequada para simulações de Poisson com números inteiros, será necessário modificá-la um pouco para utilizá-la com variáveis contínuas. O quadro da página 1180 compara as distribuições obtidas com os dois tipos de variáveis: variáveis discretas resultam em distribuições *uniformes*, enquanto variáveis contínuas resultam em distribuições *exponenciais* e distribuições *normais*.

Digite o próximo programa para transformar os números aleatórios uniformemente distribuídos, criados pela função **RND** do computador, em números distribuídos exponencialmente.

```
 10 BORDER 0: INK 7: PAPER 0:
CLS
 20 POKE 23658,0
 30 PRINT AT 0,4; INVERSE 1;"
SIMULACAO EXPONENCIAL "''
 50 INPUT "QUAL E O VALOR MEDI
O ? ";a
 70 INPUT "TAMANHO DA AMOSTRA
? ";n
 80 FOR i=1 TO n
 90 LET x=(-a)*LN (RND*1): LET
y=INT (x*10): PRINT "      ";
.1*y,
100 NEXT i
110 INPUT " OUTRA SIMULACAO (s
/n) ? ";g$
120 IF g$="s" THEN  GOTO 30
130 STOP
```

```
10 R=RND(-TIME)
30 CLS:WIDTH 40
40 PRINT TAB(8);"Simulação expo
nencial":PRINT:PRINT
50 INPUT "Qual o valor da média
";A
60 PRINT
70 INPUT "Tamanho da amostra";N
80 FOR I=1 TO N
90 X=(-A)*LOG(RND(1)):Y=INT(X*1
0):PRINT LEFT$(STR$(Y/10)+"
   ",8);
100 NEXT I
110 PRINT:INPUT "Outra vez (S/N
)";G$
120 IF G$="S" THEN 30
130 END
```

```
30  HOME
40  PRINT  TAB( 7);"SIMULACAO E
XPONENCIAL": PRINT : PRINT
50  INPUT "QUAL O VALOR DA MEDI
A ";A
60  PRINT
70  INPUT "TAMANHO DA AMOSTRA "
;N
80  FOR I = 1 TO N
90 X = ( - A) *  LOG ( RND (1))
:Y =  INT (X * 10): PRINT  LEFT
$ ( STR$ (Y / 10) + "       ",8
);
100  NEXT I
110  PRINT : INPUT "OUTRA SIMUL
ACAO (S/N) ";G$
120  IF G$ = "S" THEN 30
130  END
```

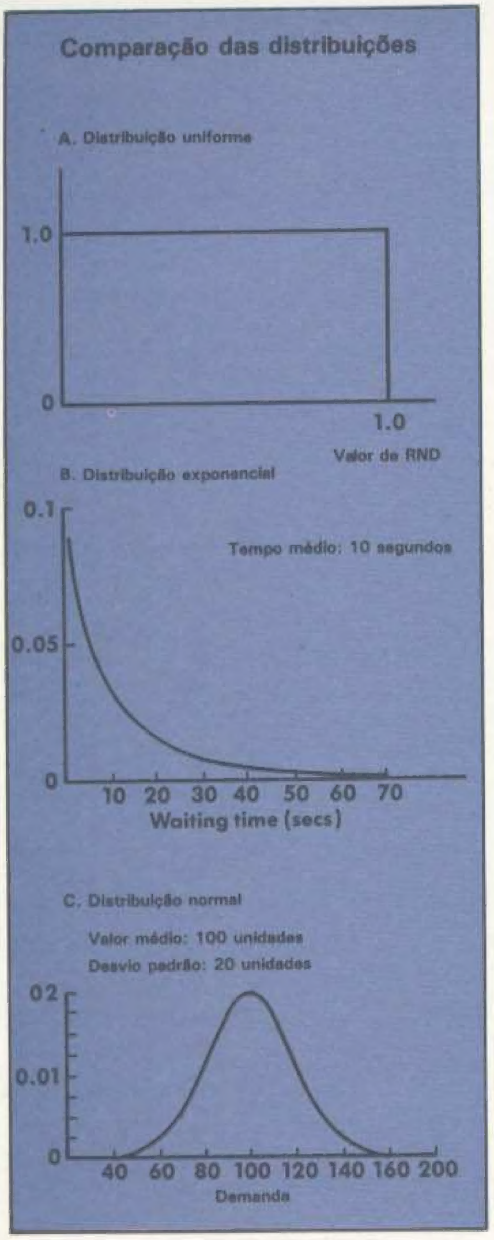
```
30 CLS
40 PRINT @5,"simulacao exponenc
ial":PRINT:PRINT
50 INPUT"QUAL E O VALOR MEDIO "
;A
60 PRINT
70 INPUT"TAMANHO DA AMOSTRA ";N
80 FOR I=1 TO N
90 X=(-A)*LOG(RND(0)):Y=INT(X*1
0):PRINT LEFT$(STR$(Y/10)+"
   ",8);
100 NEXT I
110 PRINT:INPUT"OUTRA SIMULACAO
  (S/N) ";G$
120 IF G$="S" THEN 30
130 END
```

Devemos informar ao programa, assim que o executamos, o valor da média. Feito isso, os valores da amostra são exibidos. A linha 90 realiza toda a tarefa, utilizando um recurso matemático chamado *método da transformação inversa*, para mudar a distribuição das variáveis. Esse programa pode ser usado para simular, por exemplo, o tempo de espera em um lava-rápido ou a duração das reservas de combustível em um jogo espacial. Em ambos os casos, o tempo decorrido é o elemento central.

DISTRIBUIÇÃO NORMAL

A curva normal é o tipo mais conhecido de distribuição estatística. Tem numerosas aplicações, descrevendo toda espécie de fenômeno natural — como a estatura e o peso de homens adultos. Além disso, constitui o meio mais apropriado para a representação de uma distribuição de erros.

Como exemplo do uso da distribuição normal em um modelo, consideremos esta situação: as vendas (S) de uma



empresa dependem de um valor-base (**So**) e do total gasto em propaganda (**A**). A relação poderia ser resumida na expressão **S = So + (B*A)** + desvio normal, onde **B** é uma constante conhecida.

Digite o programa a seguir para ver como funciona.

```
10 BORDER 0: PAPER 0: INK 7:
CLS
20 DIM u(50): DIM y(50): DIM
z(50)
50 PRINT AT 0,7; INVERSE 1;"
SIMULACAO NORMAL "''
60 INPUT " QUAL E O VALOR MED
IO ? ";a
70 INPUT " QUAL E O VALOR MIN
IMO ? ";m
80 INPUT " TAMANHO DA AMOSTRA
 ? ";n
90 LET b=(a-m)/2.5
100 FOR i=1 TO n
110 LET t=0
120 FOR j=1 TO 15
130 LET y(j)=RND*1
140 LET t=t+y(j): NEXT j
150 LET u(i)=((t/15)-.5)/SQR (
1/(12*15))
160 LET z(i)=a+(b*u(i))
170 LET p=INT (10*z(i)): PRINT
"     ";.1*p,
180 NEXT i
190 INPUT " OUTRA AMOSTRA ? ";
g$
200 IF g$="s" THEN  GOTO 50
210 STOP
```

```
10 R=RND(-TIME)
30 CLS:WIDTH 40
40 DIM U(50),Y(50),Z(50)
50 PRINT "Simulação normal":PRI
NT:PRINT
60 INPUT "Qual o valor da média
";A
70 INPUT "Qual o valor mínimo";
M
80 INPUT "Tamanho da amostra";N
90 B=(A-M)/2.5
100 FOR I=1 TO N
110 T=0
120 FOR J=1 TO 15
130 Y(J)=RND(1)
140 T=T+Y(J):NEXT J
150 U(I)=((T/15)-.5)/SQR(1/(12*
15))
160 Z(I)=A+(B*U(I))
170 P=INT(10*Z(I)):PRINT LEFT$(
STR$(P/10)+"        ",8);
180 NEXT I
190 PRINT:INPUT"Outra amostra";
G$
200 IF G$="S" THEN 50
210 END
```

```
30   HOME
40   DIM U(50),Y(50),Z(50)
50   PRINT  TAB( 9);"DISTRIBUICA
O NORMAL": PRINT : PRINT
60   INPUT "QUAL O VALOR DA MEDI
A ";A
70   INPUT "QUAL O VALOR MINIMO
";M
80   INPUT "TAMANHO DA AMOSTRA "
;N
90 B = (A - M) / 2.5
100   FOR I = 1 TO N
110 T = 0
120   FOR J = 1 TO 15
130 Y(J) =  RND (1)
140 T = T + Y(J): NEXT J
150 U(I) = ((T / 15) - .5) /  S
QR (1 / (12 * 15))
160 Z(I) = A + (B * U(I))
170 P =  INT (10 * Z(I)): PRINT
  LEFT$ ( STR$ (P / 10) + "
   ",8);
180   NEXT I
190   PRINT : INPUT "OUTRA  AMOS
TRA (S/N) ";G$
200   IF G$ = "S" THEN 50
210   END
```

```
30 CLS
40 DIM U(50),Y(50),Z(50)
50 PRINT @7,"distribuicao norma
l":PRINT:PRINT
60 INPUT"QUAL E O VALOR MEDIO "
;A
70 INPUT"QUAL E O VALOR MINIMO
";M
80 INPUT"TAMANHO DA AMOSTRA ";N
90 B=(A-M)/2.5
100 FOR I=1 TO N
110 T=0
120 FOR J=1 TO 15
130 Y(J)=RND(0)
140 T=T+Y(J):NEXT J
150 U(I)=((T/15)-.5)/SQR(1/(12*
15))
160 Z(I)=A+(B*U(I))
170 P=INT(10*Z(I)):PRINT LEFT$(
STR$(P/10)+"        ",8);
180 NEXT I
190 PRINT:INPUT"OUTRA AMOSTRA "
;G$
200 IF G$="S" THEN 50
210 END
```

Execute o programa e forneça o valor médio e o valor mínimo, bem como o tamanho da amostra. Tecnicamente, o intervalo de uma distribuição normal é infinito, o que deixa uma chance muito pequena de se obter um valor menor que o mínimo especificado.

As linhas 120 a 150 usam a função **RND** para criar quinze números entre 0 e 1, somando-os em seguida. A linha 160 calcula a média e o fator de escala. Depois de ter sido escalonado, o resultado é impresso.

Para maior clareza, poderá ser útil criar duas distribuições com a mesma média e aproximadamente o mesmo intervalo, mas com formatos diferentes. Execute o programa e forneça os valores 100 para a média, 50 para o mínimo e 40 para o tamanho da amostra. Se os valores obtidos fossem plotados, teríamos um gráfico semelhante ao da figura C (quadro da página 1180).

Agora, apague a linha 90 e as linhas 120 a 160. Substitua a linha 110 pela seguinte:

```
110 Z(I)=M+RND(1)*2*(A-M)
```

No TRS-Color, use **RND (0)** no lugar de **RND (1)**.

Com essas alterações, a distribuição normal é convertida em distribuição uniforme. Execute o programa novamente e entre os valores especificados, comparando os resultados. Desta vez, os valores levariam a um gráfico como o da figura A (página 1180).

Observe que a simulação normal acumula maior número de resultados próximos à média.

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR LINHA

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craf II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR NOME

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Caft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

UM LOGOTIPO PARA CADA MODELO DE COMPUTADOR

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:

Sinclair ZX-81

TRS-80

TK-2000

MSX

Spectrum

TRS-Color

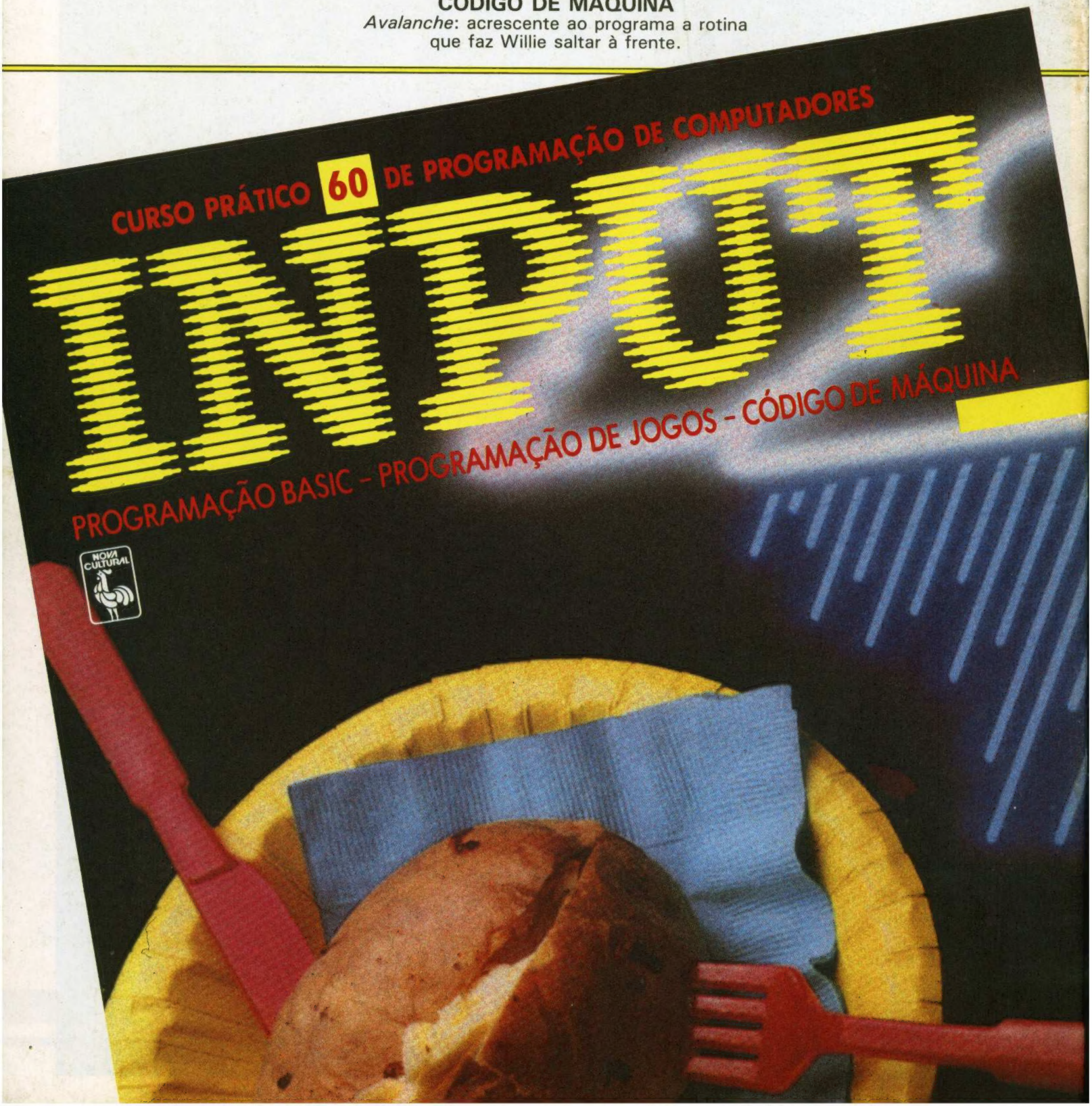
Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

NO PRÓXIMO NÚMERO

PROGRAMAÇÃO BASIC
Simulação do balanço de uma empresa. Variáveis de distribuição normal. Mercado. Controle de caixa.

PERIFÉRICOS
Funcionamento do videotexto. Informações e serviços. Conexão ao microcomputador. Compatibilidade.

PROGRAMAÇÃO BASIC
Desenhos em três dimensões. Construção do perfil. Como alterar o ângulo de visão. Rotação da figura.

CÓDIGO DE MÁQUINA
Avalanche: acrescente ao programa a rotina que faz Willie saltar à frente.